THE PRINCIPLES OF LEARNING ON MULTIPLE TASKS

Rahul Ramesh

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2025

Supervisor of Dissertation

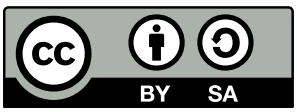
Pratik Chaudhari, Assistant Professor of Electrical and Systems Engineering

Graduate Group Chairperson

Anindya De, Associate Professor of Computer and Information Science

Dissertation Committee

Renè Vidal, Racheff Professor of Electrical and Systems Engineering and Radiology
Vijay Balasubramanian, Professor of Physics
Joshua Vogelstein, Associate Professor of Biomedical Engineering, Johns Hopkins University



ACKNOWLEDGEMENT

Write your acknowledgement text here.

ABSTRACT

THE PRINCIPLES OF LEARNING ON MULTIPLE TASKS

Rahul Ramesh

Pratik Chaudhari

Deep networks are increasingly trained on data from multiple tasks with the goal of sharing synergistic information across related tasks. Vision models, for example, are trained on over a billion images for tasks like object recognition, depth prediction and semantic segmentation. With this motivation, this dissertation studies the principles behind *how* to optimally train representations on multiple tasks and attempts to answer *why* we are able to learn representations shared across many tasks.

In the first part of the dissertation, we develop theories for training representations on multiple tasks using labeled or unlabeled data. We challenge the notion that a single pretrained representation is optimal for all tasks and show that it is optimal to instead train an ensemble of models that span the space of tasks. For labeled data, we use the lens of statistical learning theory to discuss how to: (i) split the capacity of the learner amongst related tasks; (ii) reweigh the objectives of different tasks; (iii) handle tasks that change over time. For unlabeled data, we: (i) develop a theory for self-supervised learning to train an ensemble of models that span the space of tasks; (ii) show how masked autoencoders can be adapted to different tasks by changing the scale of the noise.

The second part of this dissertation is dedicated to characterizing the nature of typical tasks, with the goal of understanding *why* representation learning works. The shocking result is that many typical tasks are highly redundant functions of the input, i.e., subspaces that vary the most and those that vary the least are both highly predictive of the outputs. We believe that this redundancy is key to understanding why we can generalize to many tasks, not just in machines, but also in organisms.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION AND OVERVIEW

1.1 Overview

Most deep networks today are trained on massive amounts of data, even if the goal is to tackle a specific task. The training data is from many different tasks as opposed to a single task, i.e., most training today is closer to the regime of multitask learning than traditional supervised learning. A vision model is pretrained on billions of images and not all of this data belongs to the same distribution as the downstream task.

This shift motivates the study of algorithms designed for multitask, meta-learning and continual learning. Although many algorithms have been developed over the last three decades (Thrun and Pratt, 1998), the key idea most popular today is identical to the one outlined nearly three decades ago in Caruana (1997) and is embarrassingly simple. To learn on multiple tasks, a single network is trained to minimize the empirical risk averaged over *all* the training data, with the weights shared across *all* the tasks. By sharing the weights across all these different tasks, the network learns a shared representation of the input.

This choice defies intuition, and yet it works incredibly well. Although learning a shared representation improves generalization (Baxter, 1995), we do not expect it to be optimal for all tasks. For example, there are many ways to represent a positive integer: binary, decimal, or using the Chinese remainder theorem. Different representations are better suited for different downstream tasks. The binary representation can be easily divided by powers of 2, and the Chinese remainder representation is convenient for multiplying two numbers. Similarly for deep networks, we expect that a single representation shared across all tasks is not statistically optimal for all tasks, and sharing the weights between dissimilar tasks can hurt generalization.

In practice, learning representations shared across all the tasks has proven to be surprisingly effective. Despite its effectiveness, this dissertation challenges the notion that this is the optimal way to learn representations. We seek to answer the following two questions: (i) What is the optimal way to train representations for many different tasks? (ii) What do typical learnable tasks look like and why can we learn a shared representation

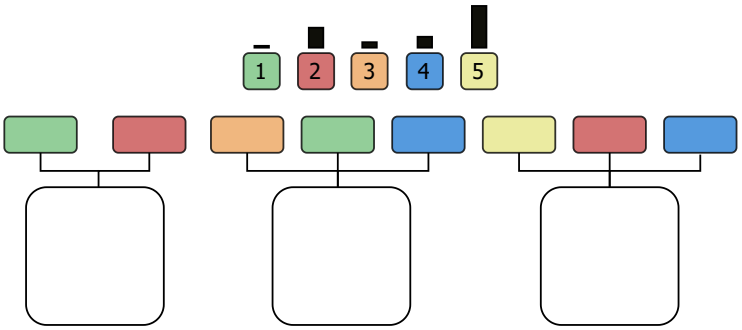


Figure 1.1: A single model is incapable of capturing the diversity of all the tasks and it is beneficial to split the capacity of the learner. In this dissertation, we develop algorithms using boosting and the theory of reference priors to train on labeled data and unlabeled data respectively.

that generalizes well to many tasks but not all of them?

1.1.1 Contributions

The first two chapters study how to train representations for multiple tasks using either labeled data or unlabeled data. The last chapter attempts to characterize the space of machine vision tasks.

In the first chapter, we use statistical learning theory to study the problem of learning from labeled data from multiple tasks. We formalize the notion of task competition and question the prevalent practice of training a single large model on all the data. We show that it is optimal to instead train multiple models on different subsets of data. We use this insight to design **Model Zoo — a collection of models trained on related subsets of the data**. Model Zoo uses a variant of boosting and is a state-of-art task-incremental continual learner. We next explore the effect of reweighing the losses of different tasks and show that the weights can be used to mitigate task competition by controlling the extent of bias and variance. We end the first chapter by introducing a **theoretical framework called prospective learning that formalizes the problem of learning from data where the distribution changes over time**. Formally, prospective learning considers the problem of generalizing to infinite future samples from a stochastic process using a finite number of samples from the past. We state the conditions under which a prospective learner that minimizes the empirical risk is able to achieve the optimal Bayes risk for a family of stochastic processes. The ideas in this chapter are rather simple, but are used in almost every single large model deployed today, either in the form of mixture of experts or through a complicated reweighing of different tasks during fine-tuning.

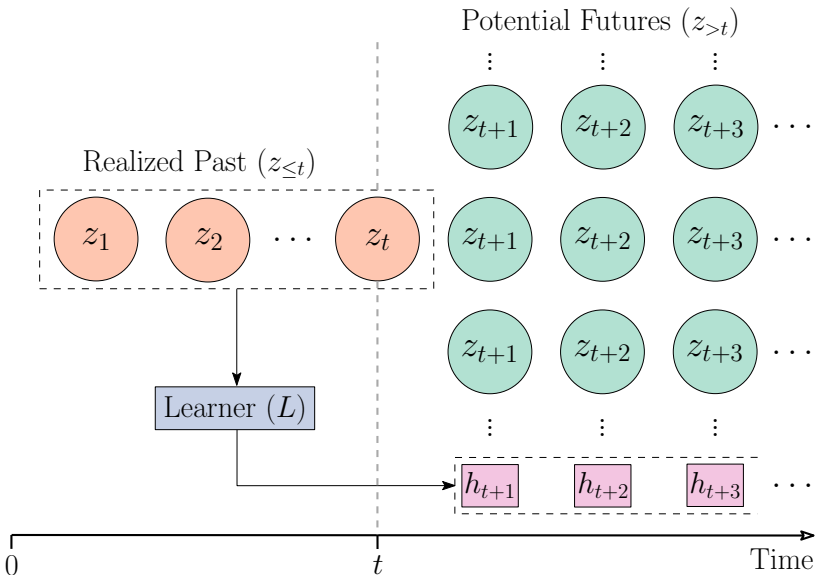


Figure 1.2: Prospective learning assumes that every sample is drawn from its own distribution, i.e., the data is drawn from a stochastic process. Generalization error is measured on samples seen in the future.

The second chapter embraces self-supervised learning as a way of pretraining representations for many downstream tasks using unlabeled data. Although most methods today train a single "foundation model" to tackle many downstream tasks, we argue entirely **from first-principles theory on reference priors that one should build an ensemble of models similar to Model Zoo**. This is the first instantiation of the concept of reference priors in machine learning, which was invented in the late 1970s. We end this chapter by showing how the objective of masked reconstruction can be adapted to different tasks — in particular the scale of the noise can be used to control the scale of the learned features. Our results on semi-supervised learning indicate that it is possible to train many smaller models using just unlabeled data and still be competitive with a large model trained on all the data.

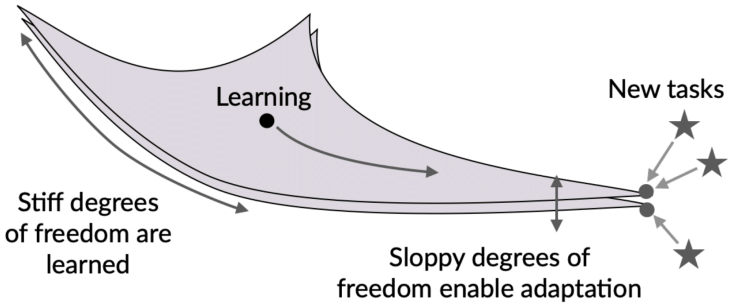


Figure 1.3: Typical tasks lie on a manifold with hyper-ribbon structure, i.e., very few directions are long and most directions are extremely thin. In other words, "typical" tasks lie close to a low-dimensional manifold.

The last chapter makes progress towards the question of *why* we are able to learn useful representations for multiple tasks. While learning theory provides answers in the asymptotic regime, the theory is far from prescriptive and the generalization bounds often vacuous. I believe that in order to make progress towards a prescriptive theory of representation learning, we must build theory specific to "typical" tasks, i.e., tasks that organisms are interested in solving and tasks that machines are made to solve. An important first step towards building this theory is to characterize the nature of typical tasks, particularly in aspects pertinent to representation learning. The two results presented in this chapter capture the phenomena that relate representation learning and the nature of typical tasks and are not explained by any existing mathematical theories.

The first result describes the manifold of the space of tasks. We show that tasks lie on a manifold with a hyper-ribbon structure, i.e., few directions are long and most other directions are incredibly thin. The second result was deduced from an attempt to explain this hyper-ribbon structure of the manifold. In particular, we find that **many typical perception tasks are highly redundant functions of the input**, i.e., directions that vary the least and directions that vary the most are both non-trivially predictive of the outputs. We find this redundancy in seen in machine vision tasks like image classification, optical flow, semantic segmentation and in auditory perception tasks like vocalization discrimination. Redundancy is incredibly shocking and forces us to rethink established ideas in neuroscience and our fundamental understanding of principal component analysis. We believe that this redundancy is important for understanding why we can share representations across lots of different tasks.

The material in this dissertation has been disseminated through the following articles:

1. **Rahul Ramesh** and Pratik Chaudhari. "Model Zoo: A Growing Brain That Learns Continually." International Conference on Learning Representations. 2021. (ICLR 21)
2. Yansong Gao*, **Rahul Ramesh***, and Pratik Chaudhari. "Deep Reference Priors: What is the best way to pretrain a model?." In International Conference on Machine Learning, pp. 7036-7051. PMLR, 2022. (ICML 22)
3. Ashwin De Silva*, **Rahul Ramesh***, Carey Priebe, Pratik Chaudhari, and Joshua T. Vogelstein. "The value of out-of-distribution data." In International Conference on Machine Learning, pp. 7366-7389. PMLR, 2023. (ICML 22)

4. **Rahul Ramesh**, Jialin Mao, Itay Griniasty, Rubing Yang, Han Kheng Teoh, Mark Transtrum, James Sethna, and Pratik Chaudhari. "A Picture of the Space of Typical Learnable Tasks." In International Conference on Machine Learning, pp. 28680-28700. PMLR, 2023. (ICML 23)

5. Jialin Mao, Itay Griniasty, Han Kheng Teoh, **Rahul Ramesh**, Rubing Yang, Mark K. Transtrum, James P. Sethna, and Pratik Chaudhari. "The training process of many deep networks explores the same low-dimensional manifold." Proceedings of the National Academy of Sciences 121, no. 12 (2024): e2310002121. (PNAS 24)

6. Ashwin De Silva*, **Rahul Ramesh***, Prospective Learning collective, Konrad Kording, Joshua Vogelstein, Pratik Chaudhari "Prospective Learning: Principled Extrapolation to the Future." In Conference on Lifelong Learning Agents, pp. 347-357. PMLR, 2023. (CoLLAs 23)

7. Ashwin De Silva*, **Rahul Ramesh***, Rubing Yang*, Siyu Yu, Joshua T. Vogelstein, Pratik Chaudhari. "Prospective Learning: Learning for a Dynamic Future." In Advances in Neural Information Processing Systems (pp. 123055–123090), 2024. (NeuRIPS 24)

8. **Rahul Ramesh***, Anthony Bisulco*, Ronald W. DiTullio, Linran Wei, Vijay Balasubramanian, Kostas Daniilidis, Pratik Chaudhari. "Many Perception Tasks are Highly Redundant Functions of their Input Data.", 2024. (Cosyne 25)

9. Anthony Bisulco*, **Rahul Ramesh***, Randall Balestriero, Pratik Chaudhari. "From Linearity to Non-Linearity: How Masked Autoencoders Capture Spatial Correlations", 2025. (Preprint)

In addition, other articles and preprints written during the Ph.D., but not included in this dissertation

1. **Rahul Ramesh**, Ekdeep Singh Lubana, Mikail Khona, Robert P. Dick, Hidenori Tanaka. "How capable can a transformer become? a study on synthetic, interpretable tasks." In International Conference on Machine Learning, 2024 (ICML 24)

2. Mikhail Khona, Maya Okawa, Jan Hula, **Rahul Ramesh**, Kento Nishi, Robert P. Dick, Ekdeep Singh Lubana, and Hidenori Tanaka. "Toward a Mechanistic Understanding of Stepwise Inference in Transformers: A Synthetic Graph Navigation Model." In International Conference on Machine Learning, 2024 (ICML 24)

3. Kento Nishi, **Rahul Ramesh**, Maya Okawa, Mikail Khona, Hidenori Tanaka, Ekdeep Singh Lubana. "Representation Shattering in Transformers: A Synthetic Study with Knowledge Editing." preprint, arXiv:2410.17194, 2024.

CHAPTER 2

STATISTICAL LEARNING THEORY FOR MULTIPLE TASKS

In this chapter, we explore how to optimally train networks, particularly when some pairs of tasks are dissimilar to each other. In this scenario, training a single network with weights shared between all tasks is not optimal. Instead, it is optimal to train a collection of models — or a Model Zoo — where each model is trained on a subset of related tasks. We conduct a comprehensive evaluation of Models Zoo and show that it achieves state-of-the-art performance in a variety of settings in task-incremental continual learning. The theory and the experimental results, despite its humble scale, force us to reconsider whether most large models today are trained optimally.

We then study task-relatedness through the lens of statistical learning theory. We find that task relatedness is highly nontrivial, that is, pairs of tasks can change from being similar to dissimilar depending on the number of samples in both tasks. We attribute this phenomenon to a bias-variance trade-off and show that reweighting the objectives of the different tasks can help us optimally use data from all tasks.

We end this chapter by defining a new theoretical framework called prospective learning to capture a broad range of scenarios — such as continual learning, transfer learning, domain generalization — where the data distribution changes over time. Prospective learning rejects the assumption that the data is separated into a discrete number of tasks, where each task is a joint distribution over the inputs and outputs. Instead, we operate under the very general assumption that the data is sampled from a stochastic process, with the goal of generalizing to future samples, using a finite number of samples from the past. We derive the theoretical conditions for asymptotic convergence to the Bayes risk and show that existing algorithms for retrospective learning fail on simple prospective learning problems.

2.1 Model Zoo: A large network vs. an ensemble of small networks

The goal of both multitask (Caruana, 1997) and task-incremental continual learning (Van de Ven et al., 2022) is to share information between tasks to better generalize on all of them. It stands to reason that a learner trained on dissimilar tasks deteriorates the models' ability to generalize. In this section, we develop techniques to explicitly split the capacity of the learner between related subsets of tasks, which encourages synergy between similar tasks while also avoiding competition between dissonant tasks. The contributions of this section are as follows:

1. **Theoretical analysis.** We characterize when it is beneficial to train a single model on multiple tasks and when doing so can be detrimental to learning. The key idea, that differentiates this work from other theoretical works on multitask learning, is to assume a notion of relatedness between different tasks. In particular, if the optimal representation for one task predicts poorly on another task, then fitting a single model on such tasks may be worse than training each task in isolation.

2. **Algorithm development.** The analysis suggests that an ideal continual learner or multitask learner benefits from splitting the capacity of the learner, between groups of synergistic tasks. Inspired by our theory, we develop Model Zoo, a collection of models trained on different subsets of tasks. This method is loosely inspired from AdaBoost in that it iteratively trains an ensemble of small models, where each learner is fit on tasks that the Zoo performs poorly on. At inference time, given the task, we average predictions from all models in the ensemble that were trained on that task.

3. **Empirical results.** We comprehensively evaluate Model Zoo on existing task-incremental continual learning benchmarks. There are a wide variety in the problem settings for continual learning, e.g., some replay data from past tasks (like Model Zoo is designed to do), some replay only a subset of data, some train only for one epoch in each 1 episode. We compare Model Zoo with existing methods in a number of these settings. Model Zoo obtains better accuracy than existing methods on the evaluated benchmarks with the improvement in average per-task accuracy being as high as 30% for Split-MiniImagenet.

4. **A critical look at continual learning.** We find that even an Isolated learner, i.e., one which trains a (small) model on tasks from each episode and does not perform any continual learning, significantly outperforms most existing continual learning methods on the evaluated benchmark problems, e.g., by more than 8% on some datasets. This strong performance is surprising because it is a very simple learner that has better training/inference time, no data replay, and a comparable number of weights to competing methods.

2.1.1 Learning a single task

In the supervised learning setting, we have access to m labeled samples $\{x_i, y_i\}_{i=1}^m$ from a distribution $P(x, y)$ where $x \in X$ and $y \in Y$. The goal is to find a hypothesis $h : X \mapsto Y$ where $h \in H$, such that it minimizes the population risk

$$e_P(h) = \mathbb{E}_{(x,y)\sim P}[h(x) \neq y].$$

We can select a hypothesis that minimizes the empirical risk

$$\hat{e}_S(h) = \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}_{\{h(x_i)\neq y_i\}}, \tag{2.1}$$

a choice motivated by the following theorem.

Theorem 1 (Vapnik (1999))**.** *Let P be any probability distribution on $X \times Y$ and let S be m samples from P. Let $D = \mathrm{VC}(H)$ be the VC-dimension of the hypothesis space. With probability at least $1 - \delta$ over the choice of samples S, if*

$$m = O\left(\frac{D - \log\delta}{\epsilon^2}\right),$$

then $e_P(h) \leq \hat{e}_S(h) + \epsilon$.

This is a classic result from statistical learning theory that assumes that the samples are independent and identically distributed. The theorem provides an upper bound for the rate at which the empirical risk of a hypothesis uniformly converges to the population risk. The sample complexity increases with the expressivity of the hypothesis class, captured by the VC-dimension D, and decreases with the desired gap between the empirical risk and the population risk denoted by ϵ.

2.1.2 Theoretical analysis of learning from multiple tasks

Multi-task (Caruana, 1997) and continual learning (Thrun, 1998; Thrun and Mitchell, 1995), consider n tasks denoted by $\bar{P} := (P_1, \ldots, P_n)$, with the corresponding training sets denoted by $\bar{S} := (S_1, \ldots, S_n) = (\{x_{1i}, y_{1i}\}_{i=1}^m, \cdots, \{x_{ni}, y_{ni}\}_{i=1}^m)$. In continual learning, specifically task-incremental continual learning, the tasks arrive sequentially, and the model observes a new task after each episode (or round); After n rounds, the learner has access to n tasks.

The goal of multitask learning is rather straightforward, which is to minimize the population risk averaged across all tasks. Continual learning, on the other hand, attempts to minimize the population risk averaged across all tasks *at every episode* — not just at the last episode. An ideal continual learner should demonstrate backward transfer or improvement on past tasks with more episodes, and forward transfer or the ability to use tasks from past episodes to improve the performance on new tasks. In this section, we analyze the population risk averaged across all tasks, with the eventual goal of developing an algorithm for both problems.

It is beneficial to learn from multiple tasks if these tasks share a set of properties which we can use to guide learning. The shared set of properties is known as an inductive bias (Thrun and Pratt, 1998), and is similar to regularization, except that it is learned using data from multiple tasks. We formalize the idea of learning an inductive bias by considering a family of hypothesis classes $\mathbb{H}$.

The family of hypothesis classes is a set containing hypothesis classes denoted by H, i.e., $H \in \mathbb{H}$ and this family is also referred to as a hyper-bias (Baxter, 1998). For example, $\mathbb{H}$ can represent a set of neural network architectures, while $H \in \mathbb{H}$ represents the set of all weights corresponding to a particular architecture. We use data from multiple tasks to select an inductive bias, which corresponds to selecting a hypothesis class H from the family of hypothesis classes $\mathbb{H}$.

The learner seeks n hypotheses $\bar{h} = (h_1, \ldots, h_n)$ where $H \in \mathbb{H}$ and $h_1, \cdots, h_n \in H$, such that it minimizes the average population risk

$$e_{\bar{P}}(\bar{h}) = \frac{1}{n} \sum_{i=1}^{n} e_{P_i}(h_i).$$

Similar to the supervised learning setting, we can select a hypothesis $\bar{h}$ that minimizes the average empirical

risk

$$\hat{e}_{\bar{S}}(\bar{h}) = \frac{1}{n}\sum_{i=1}^{n} \hat{e}_{S_i}(h_i).$$

The VC-dimension is a commonly used notion of model capacity and is cardinality of the largest set of samples that can be shattered by the hypothesis class. Baxter (1995) defines an analogous quantity for the family of hypothesis classes $\mathbb{H}$. The prediction set for samples $\bar{S}$ is defined as

$$H_{|\bar{S}|} = \left\{ \begin{bmatrix} h_1(x_{11}) & \cdots & h_1(x_{1m}) \\ \vdots & \vdots & \vdots \\ h_n(x_{n1}) & \cdots & h_n(x_{nm}) \end{bmatrix} : h_1, \cdots h_n \in H \text{ and } H \in \mathbb{H} \right\}.$$

Using the prediction set, we define the VC-dimension $d_{\mathbb{H}}(n)$, for a family of hypothesis spaces $\mathbb{H}$ to be

$$d_{\mathbb{H}}(n) := \max\{m : \max_{\bar{S}} |H_{|\bar{S}|}| = 2^{mn}\},$$

which is the largest number of samples per task for which there exists some $\bar{S}$ shattered by $\mathbb{H}$. When $n = 1$, $d_{\mathbb{H}}(n)$ is identical to the VC-dimension. Equipped with this definition, Baxter (2000) proves the following theorem.

Theorem 2 (Baxter (2000, Theorem 12))**.** *Let $\bar{P} = (P_1, \cdots, P_n)$ be n probability distributions on $X \times \{0, 1\}$ and let $\bar{S}$ be m samples per task from $\bar{P}$. Assuming that $\mathbb{H}$ is permissible, if*

$$m = O\left(\frac{1}{\epsilon^2}\left(d_H(n) - \frac{1}{n}\log d\right)\right),$$

then with probability at least $1 - \delta$ over the choice of samples $\bar{S}$, any $H \in \mathbb{H}$ and $\bar{h} \in H^n$ satisfies $e_{\bar{P}}(\bar{h}) \leq e_{\bar{S}}(\bar{h}) + \epsilon$.

The permissibility condition (Pollard (2012)) requires that $\mathbb{H}$ forms a Borel-measurable subset and that any hypothesis h that maps inputs to outputs is measurable. This is a technical condition that allows us to assign probabilities to events involving hypotheses and is almost always satisfied in practice.

Supervised learning selects a hypothesis for each task from the hypothesis space $\cup_{H \in \mathbb{H}} H$. Multitask learning

on the other hand selects a hypothesis class H from $\mathbb{H}$ before fitting a hypothesis $h \in H$ to each task. To formally argue that multitask learning is beneficial, we will analyze the conditions under which selecting a hypothesis class $H \in \mathbb{H}$, reduces the search space for the hypothesis.

More precisely, Ben-David and Borbely (2008) pointed out that

$$\text{VC}(\cup_{H \in \mathbb{H}} H) \geq d_{\mathbb{H}}(n) \geq d_{\mathbb{H}}(n+1). \tag{2.2}$$

From the first inequality, solving the n tasks together cannot be worse than solving each task in isolation. From the second inequality, the generalized VC-dimension is a non-increasing function of n. Hence, increasing the number of tasks can help reduce the sample complexity of learning each task. However, it would be fallacious to conclude that multitask learning is always beneficial for learning.

The success of multitask learning heavily relies on selecting a suitable set of hypothesis spaces $\mathbb{H}$, under which the tasks are very similar. If we pick $\mathbb{H}$ such that we are in the realizable setting, i.e., $\exists\, H^* \in \mathbb{H}$ and $\bar{h}^* \in (H^*)^n$ such that $e_{\bar{P}}(\bar{h}^*) = 0$, then theorem 2 guarantees that we asymptotically achieve low population risk. If the tasks are related, then we can construct $\mathbb{H}$ such that $H^* \in \mathbb{H}$ and $d_{\mathbb{H}}(n)$ is small, resulting in better sample complexity. Next, we present two choices of $\mathbb{H}$ and conditions under which multitask learning is beneficial.

Shared representation model. In this setup, all tasks learn a shared representation $f \in F$ and task-specific layers $\bar{g} = (g_1, \cdots, g_n) \in G^n$. The set of hypothesis spaces $\mathbb{H}$ considered here is

$$\mathbb{H} = \{H : \exists\, f \in F \text{ such that } H = G \odot f\},$$

where $f : X \mapsto V$ is a representation of the inputs and $g : V \mapsto Y$ is a classification layer.

This hypothesis is arguably the most commonly used choice in multitask learning today and outperforms a variety of architectures proposed over the last decade (Kurin et al., 2022). In particular, $f \in F$ contains most of the weights while $g \in G$ is usually just a linear layer. For this set of hypothesis spaces, Baxter (1995) proves the following theorem.

Theorem 3 (Baxter (1995))**.** *Let F and G be families of functions and let $C(\epsilon, A)$ denote the size of the smallest ϵ-cover of A. Let $\bar{P}$ denote n probability distributions on $X \times Y$, and $\bar{S}$, m samples per task from $\bar{P}$. For any $G \odot f \in \mathbb{H}$ and $\bar{h} \in G^n \times f$, if*

$$m \geq O\left(\log\left(C(\epsilon, G)\right) + \frac{1}{n}\left[\log\left(C_{e_G}(\epsilon, F)\right) - \log\delta\right]\right)$$

then $e_{\bar{P}}(\bar{h}) \leq e_{\bar{S}}(\bar{h}) + \epsilon$ with probability of at least $1 - \delta$ over draws of $\bar{S}$.

Sharing the representation layer F results in an n-fold reduction in sample complexity provided that the model capacity of the task-specific layer is small, i.e. $C(\epsilon, G) \ll C_{e_G}(\epsilon, F)$. Theorem 3 guarantees that the *average* empirical risk converges to the average population risk at a rate of $O(n^{-1/2})$. However, it does not guarantee that the empirical risk of *all tasks* converge to the population risk at the same rate of convergence; This guarantee requires a stronger assumption on the relatedness of tasks.

F-related tasks. We consider a setup where the inputs of different tasks are simple transformations of each other. A set of tasks $\bar{P}$ are F-related if for any $P, P' \in \bar{P}$

$$P' = f[P]$$

for some $f \in F$.

Let $H_{[\sim f]}$ denote a hypothesis space such that for any $h, h' \in H_{[\sim f]}$, there exists $f \in F$ such that $h' = h \odot f$. We consider the set of hypothesis spaces

$$\mathbb{H} = \{H_{[\sim f]}\}.$$

Ben-David and Borbely (2008) consider a hypothesis class H such that F acts as a group over it. In this case, each equivalence class defines a single hypothesis class $H_{[\sim f]} \in \mathbb{H}$.

We can use the following two-step algorithm to compute the optimal hypothesis (Ben-David and Borbely, 2008): (1) Find the best hypothesis class $H_{[\sim f]} \in \mathbb{H}$ to fit all tasks in $\bar{P}$. (2) Find a hypothesis $h_i \in H_{[\sim f]}$ for every P_i. The first step learns the inductive bias while the second step exploits the learned bias. We present a sample complexity bound for this algorithm below.

Theorem 4 (Ben-David and Borbely (2008))**.** *Let $\bar{P}$ denote n tasks and $\bar{S}$ denote m samples per task drawn from $\bar{P}$. Let $d_{\max} = \max_{H_{[\sim f]}} VC(H_{[\sim f]})$ and $\bar{h}^A$ be the output of the two-step algorithm. If*

$$|S_1| \geq O\left(\frac{1}{\epsilon^2}\left(2d_{max}\log\epsilon^{-1} - \log\delta\right)\right)$$

and

$$m \geq O\left(\frac{1}{\epsilon^2}\left(2d_{\mathbb{H}}\log\epsilon^{-1} - \frac{1}{n}\log\delta\right)\right)$$

then $e_{P_1}(\bar{h}_1^A) \leq e_{S_1}(\bar{h}_1^A) + \epsilon$ with a probability of at least $1-\delta$ over draws of $\bar{S}$.

Under this stronger notion of task relatedness, Ben-David and Borbely (2008) are able to upper bound the population risk of each individual task as opposed to the average population risk over all tasks like in theorem 3). The improvement in sample complexity depends on the size of the family of transformations F since it affects the size of the hypothesis class $H_{[\sim f]}$ (captured by $d_{\max}$). If we have access to data from many tasks, finding the inductive bias $H_{[\sim f]}$ requires few samples from each task. In particular, when (1) F is of finite index (2) $n \geq D\log n$ (3) $D \gg \log|F|$,. then $d_{\max} \leq d_{\mathbb{H}}(n) \ll D$, i.e., we achieve a large reduction in sample complexity as a result of multitask learning.

The above analysis assumes that all tasks are either F-related or share a low-dimensional representation. However, empirically, many datasets deteriorate in performance when all tasks are forced to share a single representation (Standley et al., 2020; Fifty et al., 2021; Ramesh and Chaudhari, 2022; Jain et al., 2023; Xie et al., 2023a), i.e., tasks can be dissimilar under the shared representation model. In the next section, we explicitly consider a model of task-relatedness to derive conditions under which training a shared representation using all the tasks isn't always optimal.

2.1.3 Task competition

Consider the set of hypothesis classes for the shared representation model, i.e., all tasks learn a shared representation $f \in F$ and a hypothesis is fit to all tasks from the hypothesis class $H_f = \{g \circ f \mid g \in G\}$. We

define

$$e_P(f) = \inf_{g \in G} e_P(g \circ f),$$

to be the infimum of the risk achievable by a particular representation. Sharing the representation f across all the tasks can sometimes be detrimental to generalization. To theoretically characterize this scenario, we consider the following notion of task-relatedness inspired by the definition in Hanneke and Kpotufe (2019).

Definition 1. *Let F be a compact space of functions and let $\mathcal{E}_P(f) = e_P(f) - \inf_{f \in F} e_P(f)$ denote the excess risk of representation $f \in F$. Two tasks P_i and P_j are ρ_{ij}-related if*

$$\mathcal{E}_{P_j}(f, f_i^*) \leq c\, \mathcal{E}_{P_i}^{1/\rho_{ij}}(f), \text{for all } f \in F. \tag{2.3}$$

where $f_i^* = \text{argmin}_{f \in F} e_{P_i}(f)$ and $\mathcal{E}_{P_j}(f, f_i^*) = \mathcal{E}_{P_j}(f) - \mathcal{E}_{P_j}(f_i^*)$

Hanneke and Kpotufe (2019) refer to ρ as the transfer exponent. Unlike Hanneke and Kpotufe (2019), we define relatedness with respect to the family of representations F as opposed to the hypothesis class. Two tasks are closely related if $\rho \in [0, \infty]$ is small. For most pairs of tasks, we expect $\rho \geq 1$; If $\rho < 1$, then task P_j has better convergence rates when using samples from P_i as opposed to samples from P_j. The definition of the transfer exponent does not assume that two tasks P_i and P_j are equally useful to each other; ρ_{ij} and ρ_{ji} can assume different values.

The transfer exponent determines how the generalization error on one task controls the generalization of another task. If a representation achieves a low generalization error on task P_i, then the same representation is also guaranteed to achieve a small generalization error on task P_j provided that ρ_{ij} is small. This definition is closely related to the inequality between tasks developed by Crammer et al. (2008) in the realizable setting, since it closely resembles a triangle inequality, i.e.,

$$e_{P_j}(f) \leq e_{P_i}(f) + e_{P_j}(f_i^*),$$

We prove the following theorem that connects our notion of task-relatedness to the generalization error.

Theorem 5 (Task Competition)**.** *Let $\bar{P}$ denote n tasks and $\bar{S}$ denote m samples per task drawn from $\bar{P}$. Let*

tasks P_i and P_j be ρ_{ij}-related according to definition 1. We arrange the tasks in increasing order of ρ_{i1}; This order is $P_{(1)}, P_{(2)}, \cdots, P_{(n)}$ and the corresponding transfer exponents are $\rho_{(1)} \leq \rho_{(2)} \leq \cdots, \rho_{(n)}$. Consider a hypothesis $\bar{h}^k = (g_1 \circ f, \cdots, h_k \circ f)$ fit on $k \leq n$ tasks. Let $\hat{h}^k$ be a hypothesis that minimizes the average empirical risk on the first $k \leq n$ tasks trained using the shared representation model. With probability at least $1 - \delta$ over draws of $\bar{S}$, the generalization error on task P_1 is bounded by

$$\mathcal{E}_{P_1}(f) \leq \frac{1}{k} \sum_{i=1}^{k} \mathcal{E}_{P_1}(f^*_{(i)}) + \frac{c}{k} \left(e_{\bar{S}}(h^k) + c' \sqrt{\frac{\log C(\epsilon_1, e_G)}{m} + \frac{\log C_{e_G}(\epsilon_2, F) - \log \delta}{km}} \right)^{1/\rho_{\max}}, \tag{2.4}$$

where $\rho_{\max}(k) = \max\left\{\rho_{(1)}, \ldots, \rho_{(k)}\right\}$, $C(\cdot, \epsilon)$ is an ϵ-cover and c, c' are constants.

Proof of theorem 5. Using the notion of task relatedness in definition 1

$$c\, \mathcal{E}^{1/\rho_{i1}}_{P_i}(f) \geq \mathcal{E}_{P_1}(f, f^*_i) = \mathcal{E}_{P_1}(f) - \mathcal{E}_{P_1}(f^*_{(i)}),$$

for any $i, j \leq n$ and $f \in F$. For convenience we denote $\rho_{(i)} = \rho_{i1}$. Summing the above equation over $i \in \{1, \ldots, k\}$ and dividing by k, we get

$$\mathcal{E}_{P_1}(f) \leq \frac{1}{k} \sum_{i=1}^{k} \mathcal{E}_{P_1}(f^*_{(i)}) + \frac{c}{k} \sum_{i=1}^{k} \mathcal{E}^{1/\rho_{(i)}}_{P_{(i)}}(f).$$

The first term measures the dissimilarity between task 1 and all the other tasks, and is the excess risk on P_1 when using f^*_i to make predictions. We bound the second term on the right-hand side to prove theorem 5. Let $\bar{P} = 1/k \sum_{i=1}^{k} P_{(i)}$. We have

$$\begin{aligned} \frac{1}{k} \sum_{i=1}^{k} \mathcal{E}^{1/\rho_{(i)}}_{P_{(i)}}(f) &\leq \frac{1}{k} \sum_{i=1}^{k} \mathcal{E}^{1/\rho_{\max}}_{P_i}(f) = \frac{1}{k} \sum_{i=1}^{k} \left(e_{P_i}(f) - e_{P_i}(f^*_i) \right)^{1/\rho_{\max}} \\ &\leq \frac{1}{k} \sum_{i=1}^{k} e^{1/\rho_{\max}}_{P_i}(f) \leq e^{1/\rho_{\max}}_{\bar{P}}(f). \end{aligned}$$

where the final step involves Jensen's inequality. This is average population risk when all tasks use the representation f. We can bound $e_{\bar{P}}(f)$ using sample complexity bounds for multitask learning (Baxter, 1995), which extend the results from Haussler (1992). The key idea is to define a cover over the family of hypothesis

spaces, which requires the family to satisfy permissibility (Pollard, 2012). Using the sample complexity bound from Baxter (1995, Theorem 1), the population risk of a hypothesis $h^k = (g_1 \circ f, \cdots, h_k \circ f)$ trained on k tasks is bounded by

$$e_{\bar{P}}(f) \leq e_{\bar{P}}(h^k) \leq e_{\bar{S}}(h^k) + c' \sqrt{\frac{\log C(\epsilon_1, e_{\mathcal{G}})}{m} + \frac{\log C_{e_{\mathcal{G}}}(\epsilon_2, \mathcal{F}) - \log \delta}{km}}$$

with probability $1 - \delta$. Putting these inequalities together,

$$\begin{aligned}
\mathcal{E}_{P_1}(f) &\leq \frac{1}{k} \sum_{i=1}^{k} \mathcal{E}_{P_1}(f^*_{(i)}) + \frac{c}{k} \sum_{i=1}^{k} \mathcal{E}^{1/\rho_{(i)}}_{P_i}(f) \\
&\leq \frac{1}{k} \sum_{i=1}^{k} \mathcal{E}_{P_1}(f^*_{(i)}) + \frac{c}{k} \left(e_{\bar{S}}(h^k) + c' \sqrt{\frac{\log C(\epsilon_1, e_{\mathcal{G}})}{m} + \frac{\log C_{e_{\mathcal{G}}}(\epsilon_2, \mathcal{F}) - \log \delta}{km}} \right)^{1/\rho_{\max}}.
\end{aligned}$$

□

Theorem 5 bounds the generalization error of every task when trained with $k \leq n$ of the most related tasks. The upper bound has two terms: the first term represents competition between the tasks and the second represents the synergy between them. The first term grows with the number of tasks k, since we add dissimilar tasks (with large $\rho_{(i)}$). The second term decreases by a factor of $k^{-1/2}$ and captures the statistical benefit of sharing weights between tasks. However, $\rho_{max}(k)$ increases with k, indicating that the samples become more inefficient as we add more tasks.

One would be tempted to overcome task competition by increasing the size of the hypothesis class. A larger hypothesis class makes it easier to find a representation f^*_i that achieves low error on tasks P_i and P_1. However, increasing the size of the hypothesis class also worsens the sample complexity of learning, indicating that there is no free lunch. If we have a finite number of samples, then splitting the capacity of the learner between related tasks is statistically optimal, particularly when tasks are dissimilar.

Figure 2.1 empirically validates our theoretical predictions on task competition. A multitask learner trained on more tasks does not lead to better accuracies on all the tasks. Most continual learning methods train a single network shared across the tasks, which forces us to accommodate dissimilar tasks. However, fig. 2.1

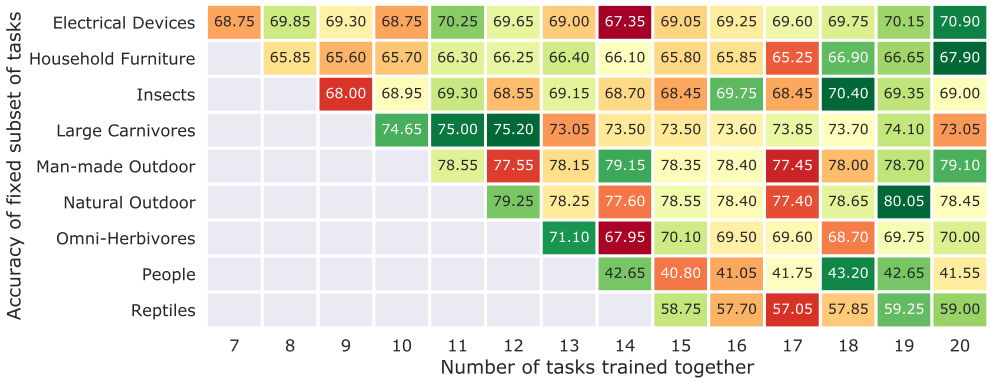


Figure 2.1: Competition between tasks can be non-trivial. We train multitask learners and track the accuracy (Y-axis) as we increasing the number of tasks used for training (X-axis) In this plot, we track the accuracy on 9 5-way classification tasks from Split-CIFAR100. A single row tracks the accuracy of a single task as we increase the number of tasks used to train the learner. Each column is a different multitask learner trained from scratch – there is no continual learning performed here. Consider the row corresponding to "Large Carnivores": an increasing number of tasks do not monotonically improve the performance on "Large Carnivores" and the accuracy drops when we include tasks #13 and #20 to the multi-task learner. Hence, there is non-trivial competition between different tasks in a dataset created from CIFAR-100 and mitigating the same is key to building better multitask and continual learners.

highlights that not all tasks are beneficial to each other. This makes continual learning challenging, since we do not have access to synergistic tasks available in the future.

Theorem 5 can be thought of as a "no free lunch theorem". It indicates that one should not always expect improved excess risk by combining data from different tasks. The three key takeaways from this result are: (1) the generalization error of a task is minimized when we train it alongside $k \leq n$ most related tasks. (2) the most related subset of tasks is different for every task. (3) the optimal choice of k balances the first and second terms in eq. (2.4). The optimal value of k is large if the tasks are similar, and small if they are dissimilar.

2.1.4 An algorithm for training Model Zoos

In the previous section, we developed theory that shows that dissimilar tasks compete for model capacity, when forced to learn a shared representation. To generalize to one of the tasks, theorem 5 tells us that it is optimal to train on a subset of the k-most related tasks. To generalize to all the tasks, we can train multiple models on different subsets of tasks, effectively splitting the capacity of the learner. Model Zoo is a simple method that instantiates this idea.

Let us assume that tasks $P_1, \ldots, P_n$ are shown sequentially to the continual learner. We assume that all tasks have the same input domain X but may have different output domains $Y_1, \ldots, Y_n$. At each "episode" k, Model Zoo is designed to train using the current task P_k and a subset of the past tasks. For example, at episode $k = 2$, we train a model with a feature generator h and task-specific classifiers to obtain models $g_1 \circ h : X \mapsto Y_1$ and $g_2 \circ h : X \mapsto Y_2$. This model can classify inputs from both tasks and outputs a probability vector $p_{g_i \circ h}(y \mid x), \forall y \in Y_i$ depending on the task. We assume that the identity of the task is known at the test time.

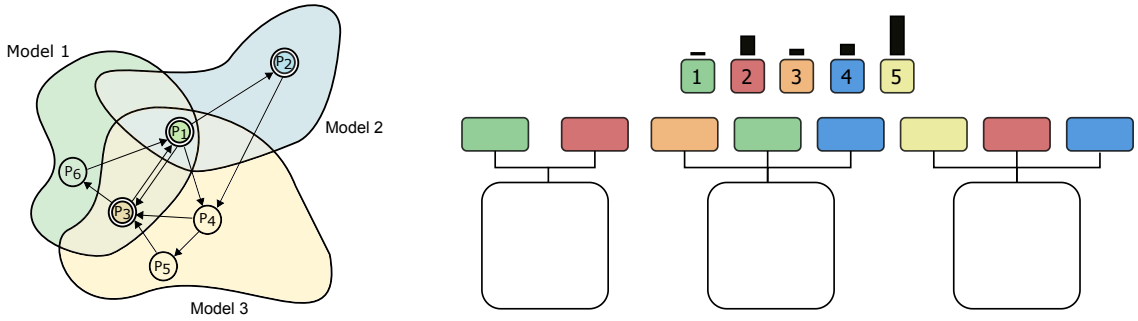


Figure 2.2: A single model is incapable of capturing the diversity of all tasks and it is beneficial to split the capacity of the learner. Ideally, we train models on synergistic tasks only; For the figure on the left, this would correspond to training Model 1 for P_1 using P_3, P_6, Model 2 for P_2 using P_1 and Model 3 for P_3 using P_1, P_4 and P_5. Model Zoo instantiates this idea and attempts to discover synergistic tasks by iteratively choosing those tasks with high loss under the ensemble.

Model Zoo adds one model to the zoo after every "episode" of continual learning. At episode k, we train the new model on samples from the newest task P_k and from $\mathscr{b} - 1$ other tasks from the set $\{P_i\}_{i=1}^{k-1}$, where $\mathscr{b}$ is the number of tasks considered for every episode of training. We favor larger values of $\mathscr{b}$ if the tasks are closely related and smaller values if tasks are dissimilar. We denote the tasks selected at episode k to be $\bar{P}_k = \{P_{\omega_k^1}, \cdots, P_{\omega_k^{\mathscr{b}}}\}$.

A model trained at episode k uses a shared representation for all tasks $\bar{P}_k$ and few task-specific classification layers. We denote the hypotheses trained on tasks $\bar{P}_k$ by $\bar{h}_k = (h_{\omega_k^1}, \cdots, h_{\omega_k^{\mathscr{b}}})$. Hypothesis $h_{\omega_k^j} : X \mapsto \mathcal{P}(Y_{\omega_k^j})$ maps the input space to a probability distribution on the output space of task $P_{\omega_k^j}$. The hypothesis can be expressed as

$$h_{\omega_k^j} = g^k_{\omega_k^j} \odot f^k$$

where f^k is the shared representation generator for model k and $g^k_{\omega_k^j}$ is the task-specific layer for task $P_{\omega_k^j}$ in

model k. We select a set of hypotheses $\bar{h}_k$ to minimize the empirical risk

$$\bar{h}_k = \operatorname*{argmin}_{\bar{h} \in H^{\flat}} e_{\bar{S}_{\omega_k}}(\bar{h}).$$

After k rounds, predictions for task P_i are the average of the outputs of all models trained on P_i, i.e.,

$$p_{k,i}(y|x) \propto \sum_{l=1}^{k} \mathbf{1}[P_i \in \bar{P}_l]\, g_i^k \odot f^k.$$

Task-specific layers like g_i^k are only viable if we know task-identities during training and inference.

Remark 6 (Task-incremental continual learning)**.** *Model Zoo operates in the task-incremental continual learning setting (Van de Ven et al., 2022); Tasks arrive sequentially and the task identities for each sample are available at both train and test times. Other variations of continual learning include class-incremental and domain-incremental continual learning (Van de Ven et al., 2022). Class-incremental continual learning tackles a single classification problem where a new class is added after every episode. Domain-incremental continual learning is similar to the task-incremental setting, except that task identities are not available at both train and test times.*

Selecting tasks for each round using boosting. Model Zoo must select $\flat$ tasks after every episode of continual learning. Ideally, it selects a related set of tasks based on the transfer exponent ρ. However, we lack access to the values ρ and estimating these distances for all pairs of tasks is expensive. Model Zoo hence uses the training loss as a proxy for task-relatedness, drawing inspiration from boosting.

AdaBoost (Schapire and Freund, 2013) builds an ensemble of weak-learners, each trained on samples which have a high loss under the ensemble. Analogously, each learner in the Model Zoo is similar to a weak-learner in AdaBoost. Each round of Model Zoo samples tasks that have high loss under the ensemble of learners. After k episodes, Model Zoo estimates the probability scores of each task $\bar{w}_k \in \mathbb{R}^k$ using

$$\bar{w}_{k,i} \propto \exp\left(-\frac{1}{m} \sum_{(x,y) \in S_i} \log p_{k,i}(y|x)\right) \tag{2.5}$$

i.e, the probability of sampling a task is proportional to its empirical loss. Model Zoo draws tasks $\bar{P}_{k+1}$ from

the multinomial distribution $\bar{w}_{k,i}$. Like AdaBoost, Model Zoo strives to achieve low errors on *all tasks*.

Model zoo is more likely to sample tasks with high empirical risk. Consider two similar tasks P_i and P_j which are dissimilar to all other tasks. We expect P_i and P_j to have a high risk when evaluated using the Model Zoo because of its dissimilarity to most other tasks. Equation (2.5) assigns P_i and P_j a higher probability in $\bar{w}_k$ in subsequent iterations of the model zoo. Hence, Model Zoo will likely train P_i and P_j together. Even if initial iterations of the zoo train on dissimilar tasks, the sampling strategy encourages future iterations of Model Zoo to train on synergistic tasks.

2.1.5 Experiments

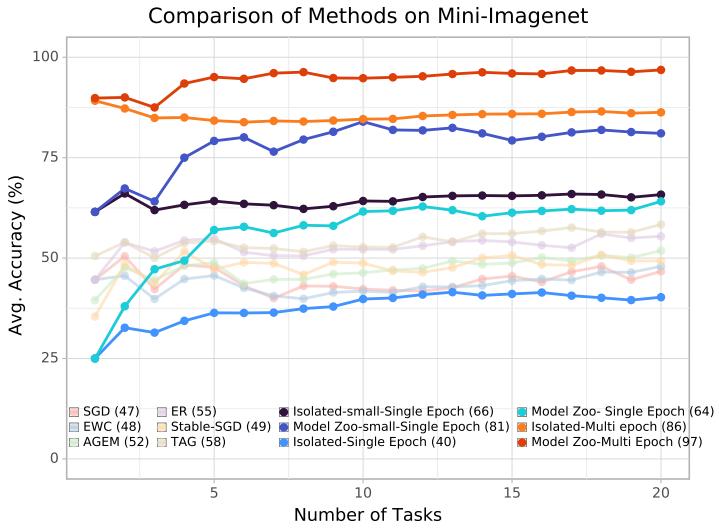


Figure 2.3: We consider the Split-MiniImagenet dataset which consists of 20 5-way classification tasks. We track the evolution of the average task accuracy of different continual learners as we increase the number of tasks seen by each learner. If the x-axis reads 7, then the continual learner has seen 7 tasks and the y-axis corresponds to the accuracy averaged over these 7 tasks. Except for the red/orange lines, we evaluate all methods in the single-epoch setting – the model trains for 1 epoch after obtaining data from a new task. The lines in bold correspond to Model Zoo and its variants while the faint lines are other single-epoch continual learning methods. "Isolated" is a variant of Model Zoo which shares no information between different tasks. "Small" indicates that each learner in the Model Zoo use a network wit 0.12M weights as opposed to a WideResnet 16-4 (3.6M weights)
Do existing methods leverage information from other tasks? Consider Isolated small-single epoch, indicated by the black line; It is a simple model that trains a small network on each task without any information shared across tasks. It has the fastest training/inference times, the smallest storage/model footprint and no data replay and yet, it outperforms other continual learning methods. This indicates *existing methods fail to leverage data from multiple tasks to improve accuracy*.
Model Zoo has an improved ability to solve each task by using other tasks. Regardless of architecture or epoch-setting, Model Zoo improves over the corresponding Isolated variant in all settings. Replaying a subset of data from past tasks alleviates catastrophic forgetting and results in forward and backward transfer.
Models in the single-epoch setting are under-trained: Isolated-single epoch (royal blue) and Isolated small-single epoch (black) differ in accuracies by 25% and Isolated-multi epoch (orange) outperforms both these methods. These two observations point to larger models being severely under-trained in the single-epoch setting. This indicates that the single-epoch setting is not appropriate for evaluating continual learning methods.

2.1.5.1 Setup

Datasets. We performed experiments using the following datasets:

(1) Rotated-MNIST (Lopez-Paz and Ranzato, 2017) uses the MNIST dataset to generate 5 different 10-way classification tasks. Each task involves using the entire MNIST dataset rotated by 0, 10, 20, 30, and 40 degrees, respectively.

(2) Permuted-MNIST (Kirkpatrick et al., 2017) involves 5 different 10-way classification tasks with each task being a different permutation of the input pixels. The first task is the original MNIST task as is convention. All other tasks are distinct random permutations of MNIST images.

(3) Split-MNIST (Zenke et al., 2017) has 5 tasks with each task consisting of 2 consecutive labels (0-1, 2-3, 4-5, 6-7, 8-9) of MNIST.

(4) Split-CIFAR10 (Zenke et al., 2017) has 5 tasks with each task consisting of 2 consecutive labels (airplane-automobile, bird-cat, deer-dog, frog-horse, ship-truck) of CIFAR10.

(5) Split-CIFAR100 (Zenke et al., 2017) has 20 tasks with each task consisting of 5 consecutive labels of CIFAR100. See the original paper for the exact constitution of each task.

(6) Coarse-CIFAR100 (Rosenbaum et al., 2017; Yoon et al., 2019) has 20 tasks with each task consisting of 5 labels. The tasks are based on an existing categorization of classes into super-classes.

(7) Split-miniImagenet (Vinyals et al., 2016) is a variant introduced in Chaudhry et al. (2019b), consisting of 20 tasks, with each task consisting of 10 consecutive labels. We merge the meta-train and meta-test categories to obtain a continual learning problem with 20 tasks. Each task containing 10 consecutive labels and 20% of the samples are used as the validation set.

The CIFAR10 and CIFAR100-based datasets consist of RGB images of size 32×32 while MNIST-based datasets consist of images of size 28×28. The miniImagenet dataset consists of RGB images of size 84×84.

Previous works have argued against using Permuted-MNIST (Prabhu et al., 2020; Farquhar and Gal, 2019b)

since it exhibits an unrealistic notion of task-relatedness. The accuracies of Coarse-CIFAR100 and Split-CIFAR100 differ by as much as 10%, suggesting that it isn't advisable to form tasks from CIFAR100 by randomly sampling classes. Random sampling also makes it harder to compare accuracies across different works.

Network architectures Benchmarks constructed from MNIST use fully-connected networks. Benchmarks based on CIFAR10 and CIFAR100 typically use the Resnet18 architecture (11.6M weights) (He et al., 2016) or a modified version of Resnet18 called Resnet18-S (1.6M weights) (Lopez-Paz and Ranzato, 2017). Model Zoo considers three different neural net architectures: 1. A wide-residual network (WRN 16-4 with 3.6M weights) 2. A small network (0.12 M) weights with 3 convolution layers 3. Resnet18-S. All networks share the network backbone and have task-specific linear classification layers.

2.1.5.2 Evaluating continual learning methods

We compare the generalization errors of all tasks across different continual learning methods. In addition, we consider other metrics like training time, inference time, model storage and sample storage. Continual learning has formulations which impose different constraints on these metrics. For example, some methods limit the model storage while others can disallow storing any samples.

It is difficult to compare methods (Prabhu et al., 2020; Farquhar and Gal, 2019a; Vogelstein et al., 2020a; Van de Ven et al., 2022) since many of them can only function in specific formulations of continual learning. We summarize some popular formulations below:

(i) Some methods (Chaudhry et al., 2019b; Guo et al., 2020; Lopez-Paz and Ranzato, 2017) chose to **store and replay data** from past tasks The amount of replay data could include all past data or a subset of it.

(ii) In the **strict formulation** (Kirkpatrick et al., 2017; Zenke et al., 2017; Kaushik et al., 2021) we store no samples from past tasks (and no replay as a result).

(iii) Some formulations enforce a **computation budget for every episode of training** (Vogelstein et al., 2020a; Ramesh and Chaudhari, 2022). For example, Lopez-Paz and Ranzato (2017); Chaudhry et al. (2019b) allow 1 epoch of training for every episode which we refer to as the **single epoch setting**.

(iv) This last setup imposes a **budget on model storage**. Methods do not evaluate on this setting formally; The constraint is implicitly imposed by the choice of architecture.

Model Zoo replays data from a subset of the past tasks with a computational budget determined by $\mathscr{b}$. It replays fewer tasks for smaller values of $\mathscr{b}$, resulting in lower training times per episode. Unlike a vanilla multi-task learner, (which has a computation budget that grows linearly with the number of episodes) the computation budget of Model Zoo is a constant across episodes, making it a legitimate continual learner. We consider the following versions of Model Zoo that abide by different formulations of continual learning:

(i) Model Zoo can access all data from past tasks. **Model Zoo (10%)** stores 10% of the samples seen at every episode and can only access these stored samples. We can compare these two models to other methods that use varying degrees of replay.

(ii) **Isolated** refers to a version of Model Zoo with $\mathscr{b} = 1$. This is identical to training a separate model on each task with no information shared across tasks. Isolated abides by the strict formulation of continual learning.

(iii) Model Zoo can reduce the required compute by decreasing the value of $\mathscr{b}$. **Model Zoo (single-epoch)** and **Isolated (single-epoch)** refer to Model Zoo and Isolated, evaluate in the single epoch setting. Strictly speaking, only Isolated (single-epoch) trains for a single epoch; Model Zoo (single-epoch) uses replay which results in more mini-batch updates. However Model Zoo (single-epoch) has comparable training times to other single-epoch methods which allows us to make fair comparisons.

(iv) Model Zoo can make use of different neural network backbones. Versions of Model Zoo that use WRN16-4, Resnet18-S and small convolution networks are called **Model Zoo**, **Model Zoo-Resnet** and **Model Zoo-Small**. We have similar versions for Isolated too. Model Zoo-Small has a comparable number of weights to architectures – like Resnet18-S – which is often used in other methods. This allows us compare Model Zoo-small to other methods when there is a constraint on model storage.

Evaluation criteria The final per-task accuracy, reported in table 2.1 and table 2.2, is the validation accuracy averaged across all tasks at the end of all episodes. The learning accuracy (Riemer et al., 2018) is the

average accuracy on a task when first seen. Higher learning accuracy indicates forward transfer – an improved ability to learn new tasks. The average per-task forgetting metric is the gap between the maximal accuracy over all episodes and the accuracy at the end, averaged across all tasks. It measures backward transfer – the ability to improve on past tasks.

2.1.5.3 Efficacy of Model Zoo as a Continual Learner

Method	Replay	Single Epoch	Rotated-MNIST	Permuted-MNIST	Split-MNIST	Split-CIFAR10	Split-CIFAR100	Coarse-CIFAR100	Split-MiniImagenet
GEM (Lopez-Paz and Ranzato, 2017)	✓	✓	86.07	82.60	-	-	67.8*	-	51.86
A-GEM (Chaudhry et al., 2019a)	✓	✓	-	89.1	-	-	62.3*	-	61.13
ER-Reservoir (Chaudhry et al., 2019b)	✓	✓	-	79.8	-	-	68.5*	-	64.03
MC-SGD (Mirzadeh et al., 2020a)	✓	✓	82.63	85.3	-	-	63.30	-	-
MEGA-II (Guo et al., 2020)	✓	✓	-	91.20	-	-	66.12	-	-
OGD (Farajtabar et al., 2020)	✗	✓	88.32	86.44	98.84	-	-	-	-
Stable-SGD (Mirzadeh et al., 2020b)	✗	✓	70.8	80.1	-	-	59.9*	-	57.79
TAG (Malviya et al., 2021)	✗	✓	-	-	-	-	62.79	-	57.2
VCL (Nguyen et al., 2018)	✓	✗	-	95.5	98.4	-	-	-	-
FRCL (Titsias et al., 2020)	✓	✗	-	94.3	97.8	-	-	-	-
FROMP (Pan et al., 2020)	✓	✗	-	94.9	99.0	-	-	-	-
EWC (Kirkpatrick et al., 2017)	✗	✗	•84	•96.9	-	-	•42.40	-	-
Prog-Nets (Rusu et al., 2016)	✗	✗	-	•93.5	-	-	•59.2	-	-
SI (Zenke et al., 2017)	✗	✗	-	•97.1	•98.9	-	-	-	-
HAT(Serra et al., 2018)	✗	✗	-	98.6	99.0	-	-	-	-
APD (Yoon et al., 2019)	✗	✗	-	-	-	-	-	56.81	-
FedWeIT (Yoon et al., 2021)	✗	✗	-	-	-	-	-	55.16	-
RMN (Kaushik et al., 2021)	✗	✗	-	97.73	99.5	-	80.01	-	-
Our methods									
Isolated-small	✗	✗	-	-	-	96.88	90.18	69.07	82.48
Model Zoo-small	✓	✗	-	-	-	96.85	92.06	73.72	94.27
Model Zoo-small (10% replay)	✓	✗	-	-	-	96.58	89.76	77.18	84.6
Isolated-Resnet	✗	✗	-	-	-	-	88.95	-	-
Model Zoo-Resnet	✓	✗	-	-	-	-	93.15	-	-
Isolated	✗	✗	99.64	98.03	99.98	97.46	91.90	80.72	86.28
Model Zoo	✓	✗	99.66	97.71	99.97	98.68	94.99	84.27	96.84
Multi-Head (multi-task)			99.66	98.16	99.98	98.11	95.38	83.19	90.83

Table 2.1: Final per-task accuracy (%) at the end of all episodes. Model Zoo outperforms other continual learning regardless of the formulation which includes: full replay (Model Zoo), limited replay (Model Zoo (10%)), no replay (Isolated) and limited model size (Model Zoo-small). Every version of Model Zoo has a higher accuracy than the corresponding version of Isolated. This indicates that Model Zoo leverages other tasks to improve the generalization error. The accuracies do not differ across architectures highlighting how Model Zoo is robust to this choice. Finally, Isolated achieves 99+% accuracy on all MNIST benchmarks which indicates it is not suitable for continual learning since tasks cannot benefit from sharing any information.
Note: * indicates that the evaluation was on Split-CIFAR100 with each task containing randomly sampled labels. • denotes that the accuracy is not from the original paper but from one of (Nguyen et al., 2018; Serra et al., 2018; Chaudhry et al., 2019a). Numbers for all methods on Split-MiniImagenet were computed using open-source implementations of the original authors.

Model Zoo outperforms **all other continual learning methods** in **all formulations of continual learning** discussed in section 2.1.5.2. Table 2.1 compares against other methods and shows that for varying degrees of replay, Model Zoo (or a relevant version of it) outperforms other methods. Table 2.2 evaluates Model

Zoo in the single-epoch setting and shows that both Model Zoo (single-epoch) and Isolated (single-epoch) outperform other methods designed for the single-epoch setting with respect to accuracy and a number of other metrics.

Model Zoo exhibits desirable traits in a continual learner like forward and backward transfer. Figure 2.4 highlights the same in the single and multi-epoch settings. Tasks seen in the past improve with more episodes indicating backward transfer. Additionally, each task starts with a better accuracy (when first seen by the learner) compared to Isolated, indicating forward transfer. These observations are further validated by table 2.2.

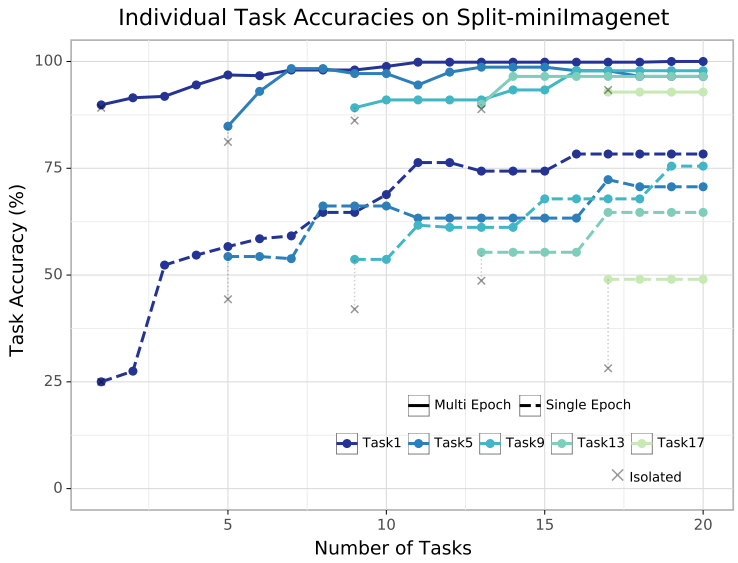


Figure 2.4: We track the accuracy of individual tasks from the Split-MiniImagenet dataset. Model Zoo shows both forward and backward transfer in this plot. Task 1 (in the single and multi-epoch settings) improves its validation accuracy with each episode, which indicates backward transfer. The 'x' denotes the accuracy of Isolated, where the task is trained without data from other tasks. The starting accuracy – the accuracy when the task is first seen by the continual learner – is higher than the accuracy of Isolated, which indicates forward transfer in Model Zoo.

In table 2.1, Model Zoo matches or outperforms the Multi-head model which is single multi-task model trained on all the tasks. Continual learning literature considers Multi-head to be the gold-standard since it has access to all tasks unlike a continual learner, which is shown tasks sequentially. It is surprising that Model Zoo achieves a 6% improvement in accuracy on Split-MiniImagenet over Multi-head. This highlights how Model Zoo avoids task competition and manages to train on synergistic groups of tasks. The Multi-head learner trains a single model on all tasks and is affected by task competition.

Model Zoo-small and Isolated-small achieve better accuracies than other algorithms indicating it is a simple and scalable method that works with different architectures. Since it uses a small convolution network, Model Zoo-small has fast training and inference times (see table 2.2) and comparable number of weights to other methods.

Method	Inference time (ms/sample)	Training time (min)	Storage Samples (%)	Storage #Weights (M)	Metrics (Multi Epoch) Accuracy (%)	Metrics (Multi Epoch) Forgetting (%)	Metrics (Multi Epoch) Forward (%)	Metrics (Single Epoch) Accuracy (%)	Metrics (Single Epoch) Forgetting (%)	Metrics (Single Epoch) Forward (%)
EWC	10.34	50	0	1.6	-	-	-	42.4	17.52	67.76
Prog-NN	-	82	0	23.7	-	-	-	59.2	0.0	59.2
GEM	10.34	1048	5–10	1.6	-	-	-	61.2	6.0	67.61
A-GEM	10.34	88	5–10	1.6	-	-	-	62.3	7.0	70.13
RMN	2712.4	-	0	11.5	80.01	-	-	-	-	-
Our methods										
Isolated-small	2.34	17.09	0	2.42	90.18	0.0	91.18	71.6	0.0	71.6
Model Zoo-small	11.70	31.71	100	2.42	92.28	0.17	90.0	73.67	0.20	71.91
Model Zoo-small (10% replay)	11.70	22.41	10	2.42	89.76	0.22	89.8	71.09	0.69	70.5
Isolated	2.34	20.76	0	54.8	91.9	0.0	91.0	50.43	0.0	50.43
Model Zoo	31.84	41.86	100	54.8	94.99	0.21	94.02	57.67	0.81	56.58

Table 2.2: We compare Model Zoo to other continual learning methods across a number of metrics. Model Zoo shows forward and backward transfer and no forgetting. It also demonstrates strong training and inference times. Model Zoo-small and Isolated-small have low model storage requirements in addition. Training times of other methods are from Chaudhry et al. (2019a) and it is the total training time in minutes for all tasks. The Inference time is the per sample prediction latency averaged over 50 mini-batches of size 16.

2.1.5.4 Rethinking the evaluation setup in continual learning

Isolated outperforms other continual learning methods as seen in tables 2.1 and 2.2; It is only outperformed by Model Zoo. This is a surprising observation since Isolated does not do any form of continual learning and does not share any information between different tasks. Consider Isolated-small in particular: It has low training/inference times (table 2.1, stores no samples for replay and uses a comparable number of weights to other methods. Hence, **Isolated is a strong continual learning baseline for most formulations of continual learning**; Recent works (Yoon et al., 2019; Liu and Liu, 2022) have made a similar observation too.

The performance of Isolated also points to another troubling conclusion. Existing methods fail to share useful information across tasks; It is preferable to train each task in isolation instead. From table 2.2, other methods have reasonable forward accuracies but all of them are significantly affected by forgetting. In an effort to combat forgetting, literature has ignored optimizing the accuracy (Díaz-Rodríguez et al., 2018). We believe it is **important build methods that outperform the Isolated baseline** and **consider replay-based**

formulations of continual learning in order to combat forgetting.

Split-MNIST, Permuted-MNIST and Rotated-MNIST are not good benchmarks for task-incremental continual learning. Since the Isolated learner achieves 99+% final per-task accuracy on all these benchmarks, there is little benefit to sharing information across tasks. Hence these benchmarks are only useful for studying model compression in a continual learning setting. Using fewer labeled samples in the training set is one alternative for MNIST-based benchmarks; This ensures that we can improve the accuracy by leveraging multiple tasks in tandem and there is some benefit to training on many tasks.

The **single-epoch setting is not conducive for evaluating continual learning methods** for two reasons: 1. Networks are severely under-trained in this regime 2. The number of epochs is not an accurate way to enforce a computational budget per episode of training. Table 2.2 and figs. 2.3 and 2.4 point to the low accuracy of single-epoch methods; The accuracy improves when trained for multiple epochs – fig. 2.3 show that the margin is as wide as 25% on Split-MiniImagenet. Models with a larger number of weights have low accuracies if allowed only 1 epoch of training; From table 2.2, the accuracy of Isolated-small (single-epoch) on Split-CIFAR100 is 23% higher than the accuracy of Isolated (single-epoch). Finally, methods like GEM take 10x longer to train than Model-Zoo even though both methods train for 1 epoch. This indicates that a single epoch of training is not the best way to enforce a per-episode limit on computation.

2.1.5.5 Ablation Studies

We study three ablation experiments to understand different components of Model Zoo. Table 2.3 shows that Model Zoo works with varying degrees of replay and works best when we use all samples from past tasks. This is unsurprising since a limited replay forces us to discard useful information. Model Zoo doesn't improve the accuracy solely due to ensemble learning; table 2.5 shows that the gains in accuracy only occur if we share information across tasks.

Table 2.4 varies the hyper-parameter ℓ and tracks the average per-task accuracy at the end of all episodes. Split-CIFAR100 improves in accuracy as we increase the number of tasks sampled per episode (ℓ) from 1 to 9. However the same monotonic trend is not observed in Split-MiniImagenet showing that task competition occurs in even common benchmark datasets.

Replay (%)	Split-CIFAR100	Split-miniImagenet
0	71.91	65.80
1	70.48	67.18
5	71.33	70.71
10	71.97	74.22
100	73.67	81.05

Table 2.3: We analyze the final per-task accuracy for varying degrees of replay. A larger replay size leads to high accuracies on both Split-CIFAR100 and Split-MiniImagenet. This also shows Model Zoo works with varying degrees of replay

# Tasks (t) (100% replay)	Split-CIFAR100	Split-MiniImagenet
1	71.91	65.02
2	72.26	67.33
5	73.67	81.05
7	73.97	88.76
9	74.13	84.9

Table 2.4: We study the final per-task accuracy for varying number of tasks sampled per episode (t). Split-CIFAR100. More tasks usually lead to better accuracies. However, this improvement is not monotonic like in Split-MiniImagenet, indicating the existence of task competition.

Method	Model Zoo	Ensemble of Isolated (100×)
Split-CIFAR100	73.67	71.46
Split-MiniImagenet	81.05	67.26

Table 2.5: We compare Model Zoo to an Ensemble of learners with no data shared across tasks. The numbers show that accuracy gains from Model Zoo are not due to ensemble learning. All ablation experiments are performed in the single-epoch setting.

2.2 Generalization error can be a non-monotonic function of amount of data

In the previous section, we developed theory to formalize task competition and showed that similar tasks benefit from being trained together. It stands to the reason that a multitask learner or or continual learner would benefit from being trained on samples from related tasks. In this section, we consider a setting where we have samples from a target task, which we would like to generalize to, and samples from a different task, which we refer to as out-of-distribution (OOD) samples. For a model trained on data from both tasks, we expect one of the following outcomes: (i) if the OOD task is similar to the target task, then more OOD samples will help us generalize to the target task; (ii) if the OOD task is dissimilar to the target task, then more samples are detrimental. In other words, we expect the generalization error to be a monotonic function of the number of OOD samples.

We show that there is a third counterintuitive possibility: the generalization error of a task can be a non-monotonic function of the number of OOD samples. As the number of OOD samples increases, the generalization error on the target task improves before deteriorating beyond a threshold. We use upper bounds for the generalization error from Ben-David et al. (2010) to show that this phenomenon arises from a bias-variance trade-off, i.e. having more OOD samples decreases the variance but increases the bias.

We first demonstrate the non-monotonic behavior through a simple but theoretically tractable problem using Fisher's Linear Discriminant (FLD). We also present empirical evidence for the presence of non-monotonic trends in target generalization error on synthetic tasks and in experimental settings constructed from MNIST, CIFAR-10, PACS and DomainNet datasets. In the setting where we know task identities of each sample, we show that non-monotonic trends can be exploited using an objective that appropriately weights the empirical risks of the target and OOD tasks.

2.2.1 Non-monotonic trends on synthetic data

We assume that the data is drawn from two tasks: n samples drawn from a target task P_t and m samples drawn from an out-of-distribution (OOD) task P_o. We would like to minimize the generalization error $e_t(h) =$

$\mathbb{E}_{(x,y)\sim P_t}\left[h(x) \neq y\right]$ on the target task. Unlike in the previous section, we assume that the entire hypothesis h is shared for both tasks and that the task identities are potentially unknown. If we were unaware of the presence of OOD samples, we would consider all data to be drawn from a single task. Therefore, we may find a hypothesis that minimizes the empirical loss

$$\hat{e}(h) = \frac{1}{n+m}\sum_{i=1}^{n+m} \ell\left(h(x_i), y_i\right), \tag{2.6}$$

using the dataset $\left\{(x_i, y_i)\right\}_{i=1}^{n+m}$; here ℓ measures the mismatch between prediction $h(x_i)$ and label y_i. If $P_t = P_o$, then $e_t(h) - \hat{e}(h) = O((n+m)^{-1/2})$ (Vapnik, 1999). But if $P_t \neq P_o$, then we should expect that error on P_t of a hypothesis obtained by minimizing the average empirical loss can be sub-optimal, especially when the number of OOD samples $m \gg n$.

2.2.1.1 An example using Fisher's Linear Discriminant

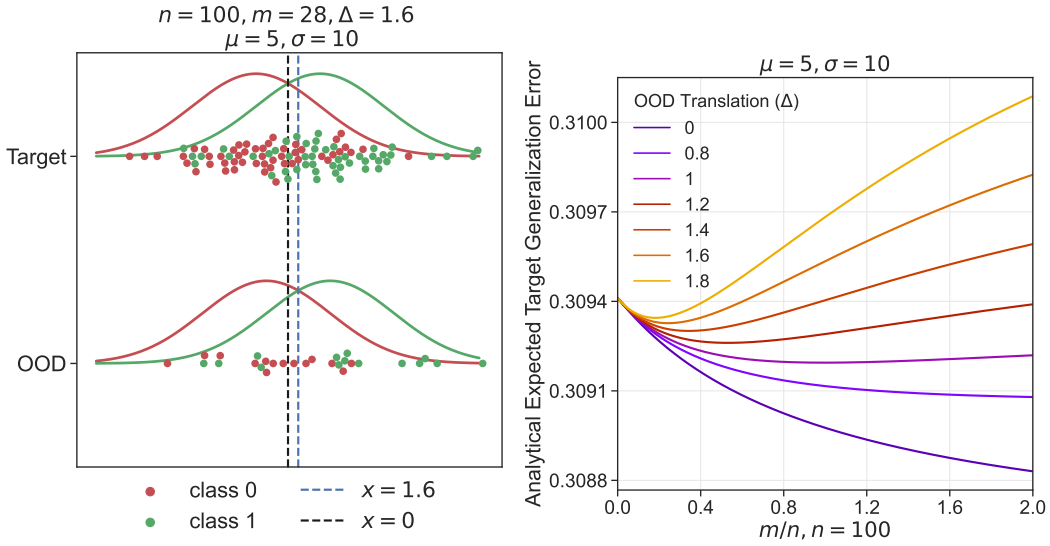


Figure 2.5: Left: A schematic of the Gaussian mixture model corresponding to the target task (top) and the OOD samples (bottom). The OOD sample size ($m = 28$) at which the target generalization error is minimized at $\Delta = 1$ is indicated at the top.
Right: For $n = 100$, we plot the generalization error of FLD on the target task as a function of the ratio of OOD and target samples m/n, for different types of OOD samples corresponding to different values of Δ. This plot uses the analytical expression for the generalization error in eq. (2.7). For small values of Δ, when the two tasks are similar to each other, the generalization error $e_t(h)$ decreases monotonically. However, beyond a certain value of Δ, the generalization error is non-monotonic in the number of OOD samples. The optimal value of m/n which leads to the best generalization error is a function of the relatedness between the two tasks, as governed by Δ in this example. This non-monotonic behavior can be explained in terms of a bias-variance trade-off with respect to the target task: a large number of OOD samples reduces the variance but also results in a bias with respect to the optimal hypothesis of the target task.

Consider a binary classification problem with one-dimensional inputs in fig. 2.5. Target samples are drawn

from a Gaussian mixture model (with means $\{-\mu, \mu\}$ for the two classes) and OOD samples are drawn from a Gaussian mixture with means $\{-\mu + \Delta, \mu + \Delta\}$. Fisher's linear discriminant (FLD) is a linear classifier for binary classification problems, and it computes $\hat{h}(x) = 1$ if $\omega^\top x > c$ and $\hat{h}(x) = 0$ otherwise; here ω is a projection vector which acts as a feature extractor, and c is a threshold that performs one-dimensional discrimination between the two classes. FLD assumes that the class conditional density of each class is a multivariate Gaussian distribution with the same covariance structure. We provide a detailed account of FLD in section B.1.1.

Suppose we fit an FLD on a dataset which comprises of n target samples and m OOD samples. Also, suppose that we do not know which samples are OOD and believe that all the samples in the dataset come from a single target distribution. For univariate data, the FLD decision rule reduces to

$$\hat{h}(x) = \begin{cases} 1, & x > \frac{\hat{\mu}_0 + \hat{\mu}_1}{2} \\ 0, & \text{otherwise.} \end{cases}$$

Define the decision threshold $\hat{c} = (\hat{\mu}_0 + \hat{\mu}_1)/2$. We can calculate (sections B.1.1 and B.1.2) an analytical expression for the generalization error of FLD on the target task:

$$e_t(\hat{h}) = \frac{1}{2}\left[\Phi\left(\frac{m\Delta - (n+m)\mu}{\sqrt{(n+m)(n+m+1)}}\right) + \Phi\left(\frac{-m\Delta - (n+m)\mu}{\sqrt{(n+m)(n+m+1)}}\right)\right]; \tag{2.7}$$

here Φ is the CDF of the standard normal distribution.

Figure 2.5 (right) shows how the generalization error $e_t(\hat{h})$ decreases up to some threshold of the ratio between the number of OOD samples and the number of samples from the target task m/n and then increases beyond that. This threshold is different for different values of Δ as can be seen in eq. (2.7) and fig. 2.5 (right). This behavior is surprising because one would *a priori* expect the generalization error to be monotonic in the number of OOD samples. The fact that a non-monotonic trend is observed even for a one-dimensional Gaussian mixture model suggests that this may be a general phenomenon. We can capture this discussion as a remark; the FLD example above is the proof.

Remark 7 (Non-monotonic generalization error)**.** *There exist target and OOD tasks, P_t and P_o respec-*

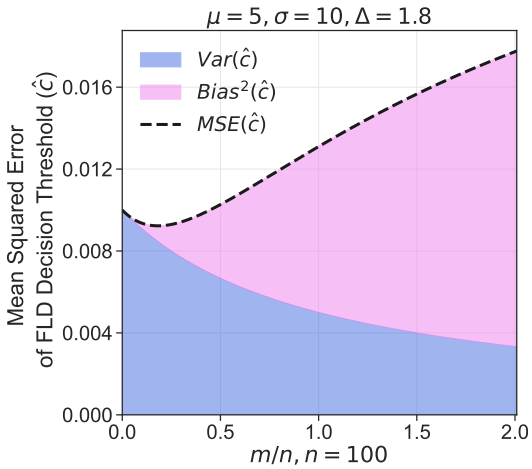


Figure 2.6: Mean squared error (MSE) (Y-axis) of the decision threshold $\hat{c}$ of FLD (see section B.1.2), for the same setup as that of fig. 2.5, plotted against the ratio of the OOD and target samples m/n (X-axis) for $\Delta = 1$. Squared bias and variance of the MSE are in violet and blue, respectively. This illustration clearly demonstrates the intuition behind non-monotonic target error: the MSE drops initially because of the smaller variance due to the OOD samples. With more OOD samples, MSE increases due to the increasing bias. Non-monotonic trend in MSE of $\hat{c}$ translates to a similar trend in the target generalization error (0-1 loss).

tively, such that the generalization error on the target task of the hypothesis that minimizes the empirical risk in eq. (2.6)*, is non-monotonic in the number of OOD samples.*

2.2.2 Non-monotonic trends for neural networks and image classification tasks

We experiment with several popular datasets including MNIST, CIFAR-10, PACS, and DomainNet and 3 different network architectures: (a) a small convolutional network with 0.12M parameters (denoted by *Small-Conv*), (b) a wide residual network (Zagoruyko and Komodakis, 2016) of depth 10 and widening factor 2 (WRN-10-2), and (c) a larger wide residual network of depth 16 and widening factor 4 (WRN-16-4).

A non-monotonic trend in generalization error can occur due to geometric and semantic nuisances. Such nuisances are very common even in curated datasets (Van Horn, 2019). We constructed 5 binary classification sub-tasks (denoted by T_i for $i = 1, \ldots, 5$) from CIFAR-10 to study this aspect (see section A.6.2). We consider a CIFAR-10 sub-task T_2 (Bird vs. Cat) as the target and introduce rotated images by a fixed angle between 0°-135°) as OOD samples. fig. 2.8 (left) shows that the generalization error decreases monotonically for small rotations but is non-monotonic for larger angles. Next, we considered the sub-task T_4 (Frog vs. Horse) as the target task and generate OOD samples by adding Gaussian blur of varying levels to images from the same task. In fig. 2.8 (middle), the generalization error on the target is a monotonically

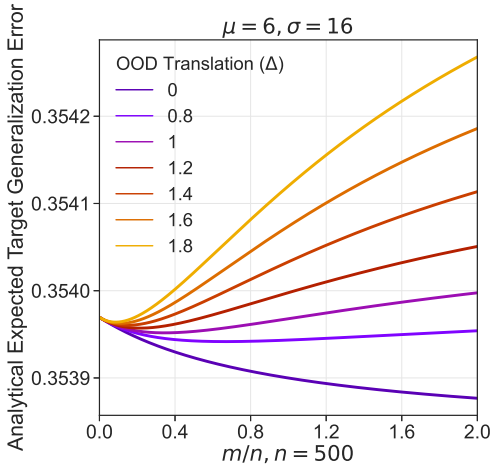


Figure 2.7: We can control the Bayes optimal error by adjusting μ, σ of the Gaussian mixture model in section 2.2.1.1. When the Bayes optimal error is large for ($\mu = 6, \sigma = 16$), we can observe non-monotonic trends even for a large number of target samples ($n = 500$). This suggests that non-monotonic trends in generalization are not limited to small sample sizes.

decreasing function of the number of OOD samples for low blur but it increases non-monotonically for high blur.

Non-monotonic trends can occur when OOD samples are drawn from a different task Large datasets can contain categories whose appearance evolves in time (e.g., a typical laptop in 2022 looks very different from that of 1992), or categories can have semantic intra-class nuisances (e.g., chairs of different shapes). We use CIFAR-10 sub-tasks to study how such differences can lead to non-monotonic trends. For 5 CIFAR-10 sub-tasks; each sub-task is a binary classification problem with two consecutive classes: Airplane vs. Automobile, Bird vs. Cat, etc. We consider (T_i, T_j) as the (target, OOD) task pair and evaluate the trend in generalization error for all 20 distinct pairs of tasks. fig. 2.8 (right) illustrates non-monotonic trends for 3 such pairs.

Non-monotonic trends also occur for benchmark domain generalization datasets We investigated three widely used benchmarks in the domain generalization literature. First, we consider the Rotated MNIST benchmark from DomainBed (Gulrajani and Lopez-Paz, 2020). We define the 10-way classification of un-rotated MNIST images as the target task and θ-rotated MNIST images as the OOD samples. Similar to the previous rotated CIFAR-10 experiment, we observe non-monotonic trends in target generalization for larger angles θ. Next, we consider the PACS benchmark from DomainBed which contains 4 distinct environments: photo, art, cartoon, and sketch. A 3-way classification task involving photos (real images) is defined as the

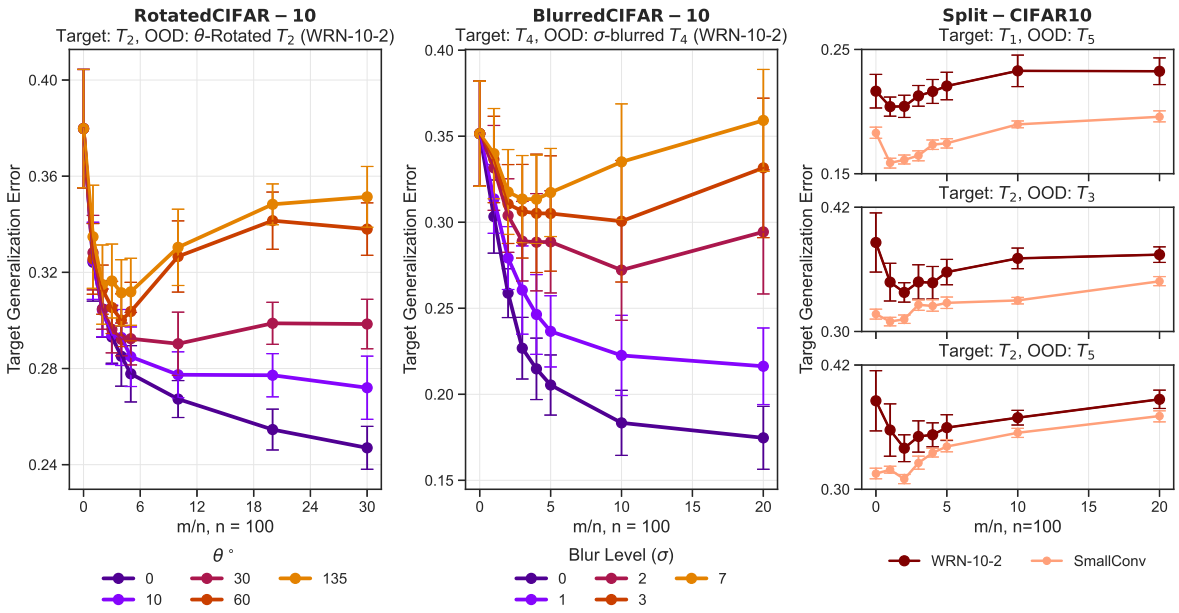


Figure 2.8: Left: Sub-task T_2 (Bird vs. Cat) from Split-CIFAR10 is the target task and images of these classes rotated by different angles θ° are the OOD task. WRN-10-2 architecture was used to train the model. We see non-monotonic curves for larger values of θ°. For 60° and 135° in particular, the generalization error at $m/n = 20$ is worse than the generalization error with a fewer OOD samples, i.e. OOD samples actively hurt generalization. **Middle:** The Split-CIFAR10 binary sub-task T_4 (Frog vs. Horse) is the target task and images with different levels of Gaussian blur are the OOD samples. WRN-10-2 architecture was used to train the model. Non-monotonic curves are observed for larger levels of blur, while for smaller levels of blur, we notice that adding more OOD data improves the generalization on the target task. **Right:** Generalization error of two separate networks, WRN-10-2 and SmallConv, on the target task is plotted against the number of samples from the OOD task for 3 different pairs of target-OOD tasks from Split-CIFAR10. All the 3 pairs exhibit non-monotonic target generalization trends across both network models. Error bars indicate 95% confidence intervals (10 runs).

target task, and we let the corresponding data from other environments be the OOD samples. Interestingly, we observe that when OOD samples consist of sketched images, then the generalization error on the real images exhibits a non-monotonic trend. We also observe similar trends in DomainNet, a benchmark that resembles PACS; see fig. 2.9.

Generalization error is not always non-monotonic even when there is distribution shift We considered CINIC-10 (Darlow et al., 2018), a dataset which was created by combining CIFAR-10 with images selected and down-sampled from ImageNet. We train a network on a subset of CINIC-10 that comprises of both CIFAR-10 and ImageNet images. The target task is CIFAR-10 itself, so images from ImageNet in CINIC-10 act as OOD samples. fig. 2.10 demonstrates that having more ImageNet samples in the training data improves the generalization (monotonic decrease) on the target task, but at a slower rate than the instance where the training data is purely comprised of target data. This phenomenon is also demonstrated in fig. 2.5: for

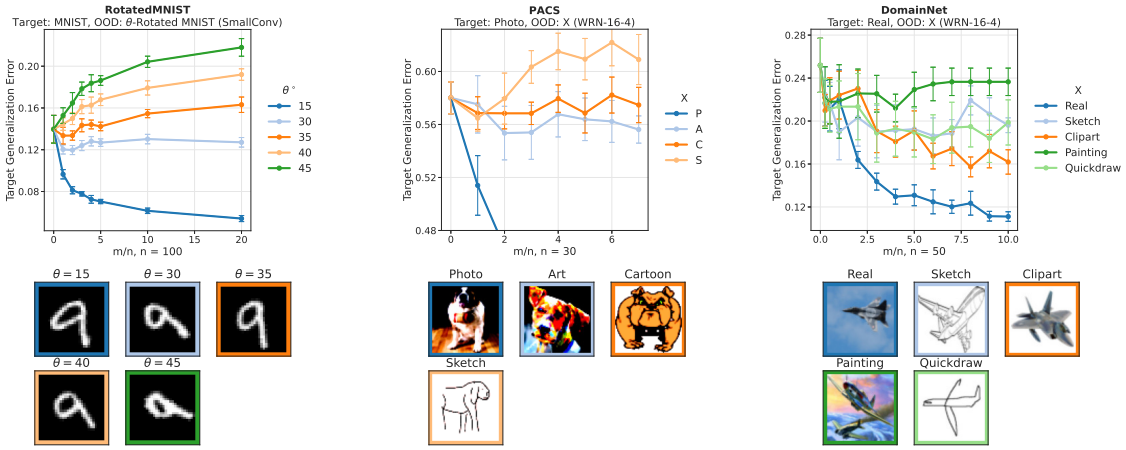


Figure 2.9: Non-monotonic trends in target generalization error on three DomainBed benchmarks. Left: Rotated MNIST (10 classes, 10 target samples/class, *SmallConv*), Middle: PACS (3 classes {dog, elephant, horse}, 10 target samples/class, WRN-16-4), and Right: DomainNet (2 classes {bird, plane}, 25 target samples/class, WRN-16-4). Error bars indicate 95% confidence intervals (10 runs).

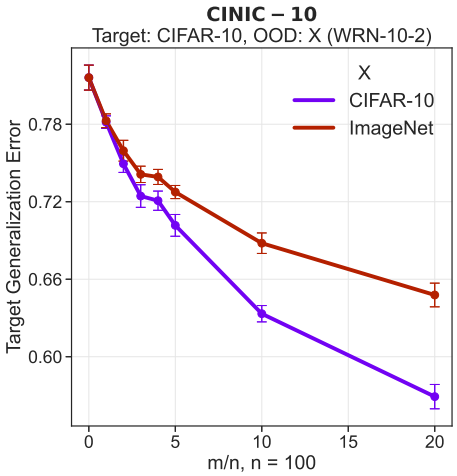


Figure 2.10: Target task is CIFAR-10 and OOD samples are from ImageNet. Although there is a distribution shift that causes the red curve to be higher error than the purple one, there is no non-monotonic trend in the generalization on CIFAR-10 due to OOD samples from ImageNet. Error bars indicate 95% confidence intervals (10 runs).

sufficiently small shifts, the target generalization error decreases as the number of OOD samples increases.

2.2.3 Weighted objective to optimally exploit all tasks

Assumption in Sections 3.1 and 3.2 In the previous section, we discussed non-monotonic trends in generalization error due to the presence of OOD samples in training datasets. If we do not know which samples are OOD, then the generalization for the intended target task can deteriorate. But it is statistically challenging

to identify which samples are OOD. We neither propose nor use an explicit method to do this in this section. Instead, we assume for the sake of analysis that the identities of the target and OOD samples in the datasets are known in advance. We begin by stating the following theorem.

Theorem 8 (Ben-David et al. (2010))**.** *For two tasks P_t and P_o, let $\hat{h}_\alpha$ be the minimizer of the α-weighted empirical risk $\hat{e}_\alpha(h) = \alpha\hat{e}_t(h) + (1-\alpha)\hat{e}_o(h)$ where $\hat{e}_t(h)$ and $\hat{e}_o(h)$ are the empirical risks of P_t and P_o respectively. The generalization error*

$$e_t(\hat{h}_\alpha) \leq e_t(h_t^*) + 4\sqrt{\left(\frac{\alpha^2}{n} + \frac{(1-\alpha)^2}{m}\right)\left(V_H \log(2(m+n+1)) + 2\log\frac{8}{\delta}\right)} + 2(1-\alpha)(\frac{1}{2}d_H(P_t, P_o) + \lambda),$$

with probability at least $1-d$. Here $h_t^* = \operatorname{argmin}_{h\in H} e_t(h)$ is the target error minimizer; V_H is the VC-dimension of the hypothesis class H, $\lambda = e_t(h^*) + e_{\text{ood}}(h^*)$ and $d_H(P_t, P_o)$ is the $H\Delta H$-divergence which is a notion of relatedness between the tasks P_t and P_o.

The theorem shares a lot of similarity to the task competition theorem (theorem 5), in that it also captures a bias-variance trade-off. However, instead of splitting the model capacity, we consider reweighing the losses of different tasks using weights α. If we use an appropriate value of α that makes the second and third terms on the right-hand side small, then we can mitigate the deterioration of generalization error due to OOD samples. If the OOD samples are very different from those of the target task, i.e., if $d(P_t, P_o)$ is large, then this theorem suggests that we should pick an $\alpha \approx 1$. Doing so effectively ignores the OOD samples and the generalization error then decreases monotonically as $O(n^{-1/2})$. Note that computation and minimization of the α-weighted convex combination of target and OOD losses, $\alpha\hat{e}_t(h) + (1-\alpha)\hat{e}_o(h)$, is possible *only* when the identities of target and OOD samples are known in advance.

2.2.3.1 Choosing the optimal α^*

If we define $\rho = \frac{\sqrt{V_H - \log d}}{d_H(P_t, P_o)}$ to be, roughly speaking, the ratio of the capacity of the hypothesis class and the distance between tasks, then a short calculation shows that for $\alpha \in [0, 1]$,

$$\alpha^* = \begin{cases} 1 & \text{if } n \geq 4\rho^2, \\ \frac{n}{n+m}\left(1 + \sqrt{\frac{m^2}{4\rho^2(n+m) - nm}}\right) & \text{else.} \end{cases}$$

This suggests that if we have a hypothesis space with small VC-dimension or if the OOD samples and target samples come from very different distributions, then we should train only on the target samples to obtain optimal error. Otherwise, including the OOD samples after appropriately weighing them using α^* can give a better generalization error.

Estimating ρ is difficult because it depends on the VC-dimension of the hypothesis class (Ben-David et al., 2010; Vedantam et al., 2021). But in general, we can treat α as a hyperparameter and use validation data to search for its optimal value. For our FLD example we can do slightly better: we can calculate the analytical expression for the generalization error for the hypothesis that minimizes the α-weighted empirical risk (see sections B.1.3 and B.1.4) and calculate α^* by numerically evaluating the expression for $\alpha \in [0, 1]$.

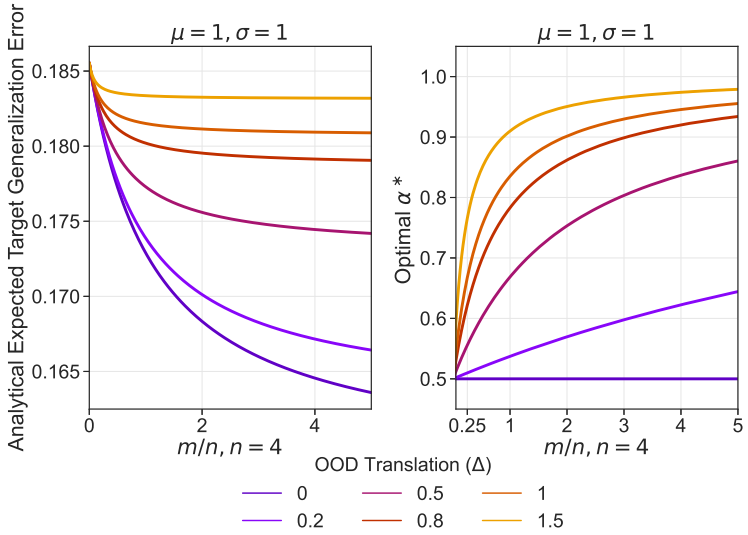


Figure 2.11: Left: Generalization error on the target task for the Gaussian mixture model using a weighted objective (theorem 8) in FLD; see section B.1.3. Note that unlike in fig. 2.5, the generalization error monotonically decreases with the number of OOD samples m. **Right:** The optimal α^* that yields the smallest target generalization error as a function of the number of OOD samples. Note that α^* increases as the number of OOD samples m increases; this increase is more drastic for large values of Δ and is more gradual for small values of Δ. Observe that $\alpha^* = 1/2$ for all values of m if $\Delta = 0$.

Figure 2.11 shows that regardless of the number of OOD samples (m) and the relatedness between OOD and target tasks (Δ), we can obtain a generalization error that is always better than that of a hypothesis trained without OOD samples. In other words, if we choose α^* appropriately (fig. 2.5 corresponds to choosing $\alpha = 1/2$), then we do not suffer from non-monotonic generalization error on the target task.

2.2.3.2 Training networks with the α-weighted objective

In section 2.2.2, for a variety of computer vision datasets, we found that for some pairs of tasks, the generalization error is non-monotonic in the number of OOD samples. We now show that if we knew which samples were OOD, then we can rectify this trend using an appropriate value of α^* to weigh the samples differently. In fig. 2.12, we track the test error of the target task for three cases: training is agnostic to the presence of OOD samples (red), the learner knows which samples are OOD and uses an $\alpha = 1/2$ in the weighted risk to train (yellow, we call this "naive"), and when it uses an optimal value of α using grid-search (green). Searching over α improves the test error on all these 3 pairs of target-OOD tasks.

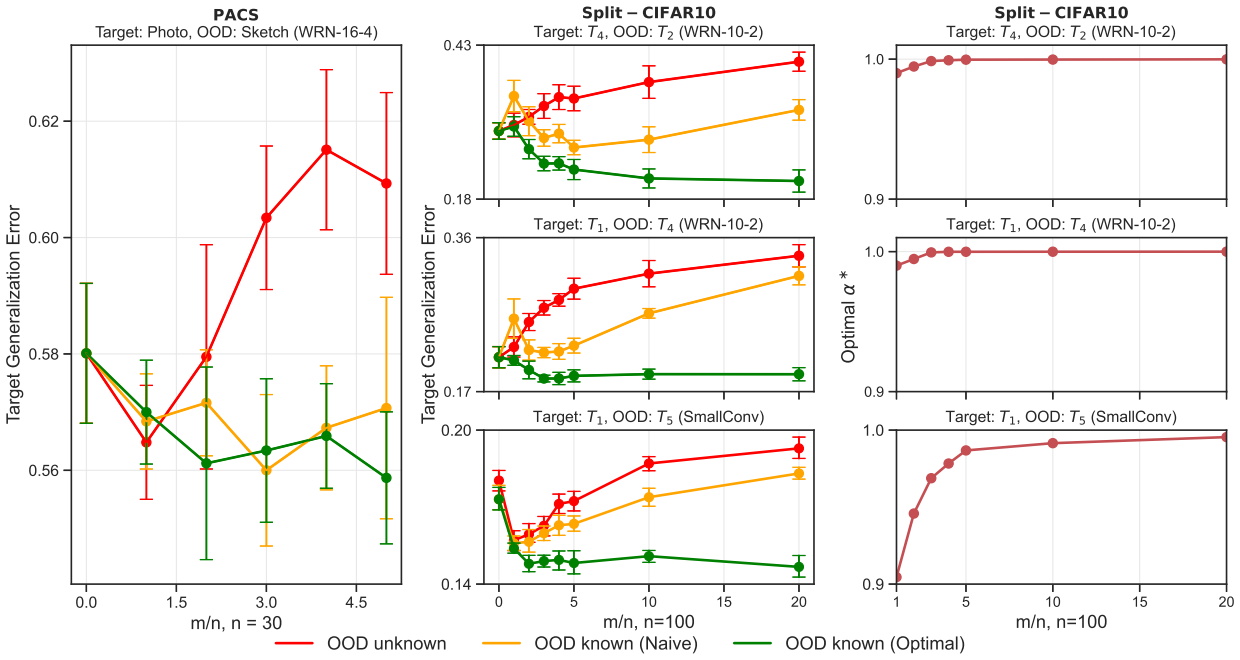


Figure 2.12: Here we present three settings: minimizing the average risk over target and OOD samples is agnostic to OOD samples present (red), minimizing the sum of the average loss of the target and OOD tasks which corresponds to $\alpha = 1/2$ (yellow), minimizing an optimally weighted convex combination of the target and OOD empirical loss (green). The last two settings are only possible when one knows which samples are OOD. For each setting, we plot the generalization error on the target task against the number of OOD samples for (target, OOD) pairs from PACS **(Left)** and CIFAR-10 subtasks **(Middle)**. Unlike in CIFAR-10 task pairs, we observe that in PACS, the target generalization error has a downward trend when $\alpha = 0.5$ (yellow line, left panel). We speculate that this could be due to the similarity between the target and OOD tasks, which causes the model to generalize to the target even at a naive weight. **Right:** The optimal α^* obtained via grid search for the three problems in the middle column plotted against different number of OOD samples. The value of α^* lies very close to 1 but it is never exactly 1. In other words, if we use the weighted objective in theorem 8 then we always obtain some benefit, even if it is marginal when OOD samples are very different from those of the target. Error bars indicate 95% confidence intervals over 10 experiments.

Sampling mini-batches during training For $m \gg n$, mini-batches that are sampled uniformly randomly from the dataset will be dominated by OOD samples. As a result, the gradient even if it is still unbiased, is

computed using very few samples from the target task. This leads to an increase in the test error, which is particularly noticeable with α^* chosen appropriately after grid search. We therefore use a biased sampling procedure where each mini-batch contains a fraction β samples from the target task and the remainder $1 - \beta$ consists of OOD samples. This parameter controls the bias and variance of the gradient of the target task ($\beta = \frac{n}{n+m}$ gives unbiased gradients with respect to the unweighted total objective and high variance with respect to the target task when $m \gg n$. We found that both $\beta = \{0.5, 0.75\}$ improve test error.

Weighted objective for over-parameterized networks It has been argued previously that weighted objectives are not effective for over-parameterized models such as deep networks because both surrogate losses $\hat{e}_t(h)$ and $\hat{e}_o(h)$ are zero when the model fits the training dataset (Byrd and Lipton, 2019). It may therefore seem that the weighted objective in theorem 8 cannot help us mitigate the non-monotonic nature of the generalization error; indeed the minimizer of $\alpha\hat{e}_t(h) + (1 - \alpha)\hat{e}_o(h)$ is the same for any α if the minimum is exactly zero. Our experiments suggest otherwise: the value of α does impact the generalization error—even for deep networks. This is perhaps because even if the cross-entropy loss is near-zero for a deep network towards the end of training, it is never exactly zero.

Limitations of using a weighted objective The numerical and experimental evidence above indicate that even a weighted empirical risk minimization (ERM) algorithm between the target and OOD samples is able to rectify the non-monotonicity. However, this procedure is dependent on two critical ideal conditions: (1) We must know which samples in the dataset are OOD, and (2) We must have a held out dataset of target samples to tune the weight α. The difficulty of meeting both of these conditions in reality limits the utility of this procedure as a practical solution to the problem. Instead, we hope that it would serve as a proof-of-concept solution that motivates future research into accurately identifying OOD samples within datasets and designing ways of determining the optimal weights.

2.3 Prospective learning: A framework for learning over time

We end this chapter by presenting a new framework for learning from multiple tasks over time. In the first two sections, we assume that each sample belongs to one of a finite number of tasks and that the samples from every task are independent and identically distributed. Prospective learning is a theoretical framework that takes this assumption to its limit and assumes that every sample belongs to its own task.

In PAC-learning, the samples are independent and identically distributed, i.e., the future is identical to this past. This assumption is neither testable nor believed to be true in practice. The future is always different from the past and those changes may cause the optimal hypothesis to change over time as well. There are numerous mathematical and empirical approaches that have been developed to address this issue, e.g., techniques for being invariant to, or adapting to, distribution shift, modeling the future as a different task, etc. But we lack a first-principles framework to address problems where data distributions and goals may change over time in such a way that the optimal hypothesis is time-dependent.

This section develops a theoretical framework called "Prospective Learning" (PL). Instead of data arising from an unknown probability distribution like in PAC learning, prospective learning assumes that data is drawn from an unknown stochastic process, and that the optimal hypothesis may change over time. A prospective learner uses samples received up to some time $t \in \mathbb{N}$ to output an infinite sequence of predictors, which it uses for making predictions on data at all future times $t' > t$.

Why should one care about prospective learning? Imagine a deployed machine learning system. The designer of this system desires to optimize—not the risk upon the past training data, or the risk on the immediate future data—but the risk on all data that the model will make predictions upon in the future. As data evolves, e.g., due to changing trends and preferences of the users, the optimal hypothesis to make predictions also changes. Time is the critical piece of information if the system designer is to achieve their goals. Both in the sense of how far back in time a particular datum was recorded, and in the sense of how far ahead in the future this system will be used to make predictions. The designer must take time into account to avoid retraining the model periodically, *ad infinitum*.

2.3.1 A definition of prospective learning

A prospective learner minimizes the expected cumulative risk of the future using past data. Such a learner is defined by the following key ingredients (see fig. 2.13 (left) for schematic illustration).

Data. Let the input and output at time t be denoted by $x_t \in \mathcal{X}$ and $y_t \in \mathcal{Y}$ respectively. Let $z_t = (x_t, y_t)$. We will model the data as a stochastic process $Z \equiv (Z_t)_{t\in\mathbb{N}}$ defined on an appropriate probability space $(\Omega, \mathcal{F}, \mathbb{P})$. At time $t \in \mathbb{N}$, denote past data by $z_{\leq t} \equiv (z_1, \dots, z_t)$ and future data by $z_{>t} \equiv (z_{t+1}, \dots)$. We will find it useful to distinguish between the realization of the data, denoted by $z_{\leq t}$, and the corresponding random variable, $Z_{\leq t}$.

Hypothesis class. At time t, a prospective learner selects an infinite sequence $h \equiv (h_1, \dots, h_t, h_{t+1}, \dots)$ which it uses to make predictions on data at any time in the future. Each element of this sequence $h_t : \mathcal{X} \mapsto \mathcal{Y}$ and therefore $h_t \in \mathcal{Y}^{\mathcal{X}}$.[1] The hypothesis class $\mathcal{H}$ is the space of such hypotheses, $h \in \mathcal{H} \subseteq (\mathcal{Y}^{\mathcal{X}})^{\mathbb{N}}$.[2] We will again use the shorthand $h_{\leq t} \equiv (h_1, \dots, h_t)$. We will sometimes talk about a "time-agnostic hypothesis" which will refer to a hypothesis such that $h_t = h_{t'}$ for all $t, t' \in \mathbb{N}$. Observe that this makes our setup different from the standard setup in PAC learning where the learner selects a single hypothesis in $\mathcal{Y}^{\mathcal{X}}$. One could also think of prospective learning as using a single time-varying hypothesis $h : \mathbb{N} \times \mathcal{X} \mapsto \mathcal{Y}$, i.e., the hypothesis takes both time and the datum as input to make a prediction.

Learner. A prospective learner is a map from the data received up to time t, to a hypothesis that makes predictions on the data over all time (past and future): $(\mathcal{X} \times \mathcal{Y})^{\dagger} \to (\mathcal{Y}^{\mathcal{X}})^{\mathbb{N}}$. The learner gives as output a hypothesis $h(z_{\leq t}) \in \mathcal{H}$. Unlike a PAC learner, a prospective learner can make different kinds of predictions at different times. This is a crucial property of prospective learning. In other words, after receiving data up to time t, the hypothesis selected by the prospective learner can predict on samples at any future time $t' > t$.

[1] We will use some non-standard notation in this paper. In particular, a hypothesis h will always refer to sequence of predictors $h \equiv (h_1, \dots, h_t, h_{t+1}, \dots)$. This helps us avoid excessively verbose mathematical expressions.

[2] When we say that "learner selects a hypothesis" in the sequel, it will always mean that the learner selects an infinite sequence from within the hypothesis class $\mathcal{H}$.

Prospective loss and risk. The future loss incurred by a hypothesis h is

$$\bar{\ell}_t(h, Z) = \limsup_{\tau \to \infty} \frac{1}{\tau} \sum_{s=t+1}^{t+\tau} \ell(s, h_s(X_s), Y_s), \tag{2.8}$$

where $\ell : \mathbb{N} \times \mathcal{Y} \times \mathcal{Y} \mapsto [0, 1]$ is a bounded loss function. [3] Prospective risk at time t is the expected future loss

$$R_t(h) = \mathbb{E}\left[\bar{\ell}_t(h, Z) \,|\, z_{\leq t}\right] = \int \bar{\ell}_t(h, Z) \, \mathrm{d}\, \mathbb{P}_{Z \,|\, z_{\leq t}}, \tag{2.9}$$

where we assume that h is a random variable and $h \in \sigma(Z_{\leq t})$ where $\sigma(\cdot)$ denotes the filtration (an increasing sequence of sigma algebras) of the stochastic process Z. We have used the shorthand $\mathbb{E}[Y \,|\, x]$ for $\mathbb{E}[Y \,|\, X = x]$. Observe that we have conditioned the prospective risk of the hypothesis h upon the realized data $z_{\leq t}$. We can take an expectation over the realized data, to obtain the expected prospective risk

$$\mathbb{E}\left[R_t(h)\right] = \int R_t(h) \, \mathrm{d}\, \mathbb{P}_{Z_{\leq t}}.$$

Prospective Bayes risk is the minimum risk achievable by any hypothesis. In PAC learning, it is a constant that depends upon the (fixed) true distribution of the data and the risk function. In prospective learning, the optimal hypothesis can predict differently at different times. We therefore define the prospective Bayes risk at a time t as

$$R_t^* = \inf_{h \in \sigma(Z_{\leq t})} R_t(h), \tag{2.10}$$

which is the minimum achievable prospective risk by any learner that observes data $z_{\leq t}$. We define the Bayes optimal learner as any learner that achieves a Bayes optimal risk at every time $t \in \mathbb{N}$. In certain contexts, one might be interested in the limiting prospective Bayes risk as $t \to \infty$.

2.3.1.1 Different prospective learning scenarios with illustrative examples

We next discuss four prospective learning scenarios that are relevant to increasingly more general classes of stochastic processes. Our goal is to illustrate, using examples, how the definitions developed in the previous section capture these scenarios. We will assume that for all times t we have $X_t = 1, Y_t \in \{0, 1\}$. We will also

[3] The limsup is guaranteed to exist if ℓ is bounded. If the series always converges, we can use lim instead.

focus on the time-invariant zero-one loss $\ell(t, \hat{y}, y) = \delta(\hat{y} \neq y)$ for all t, here δ is the Dirac delta function. Figure 2.13 shows example realizations of the data for each scenario.

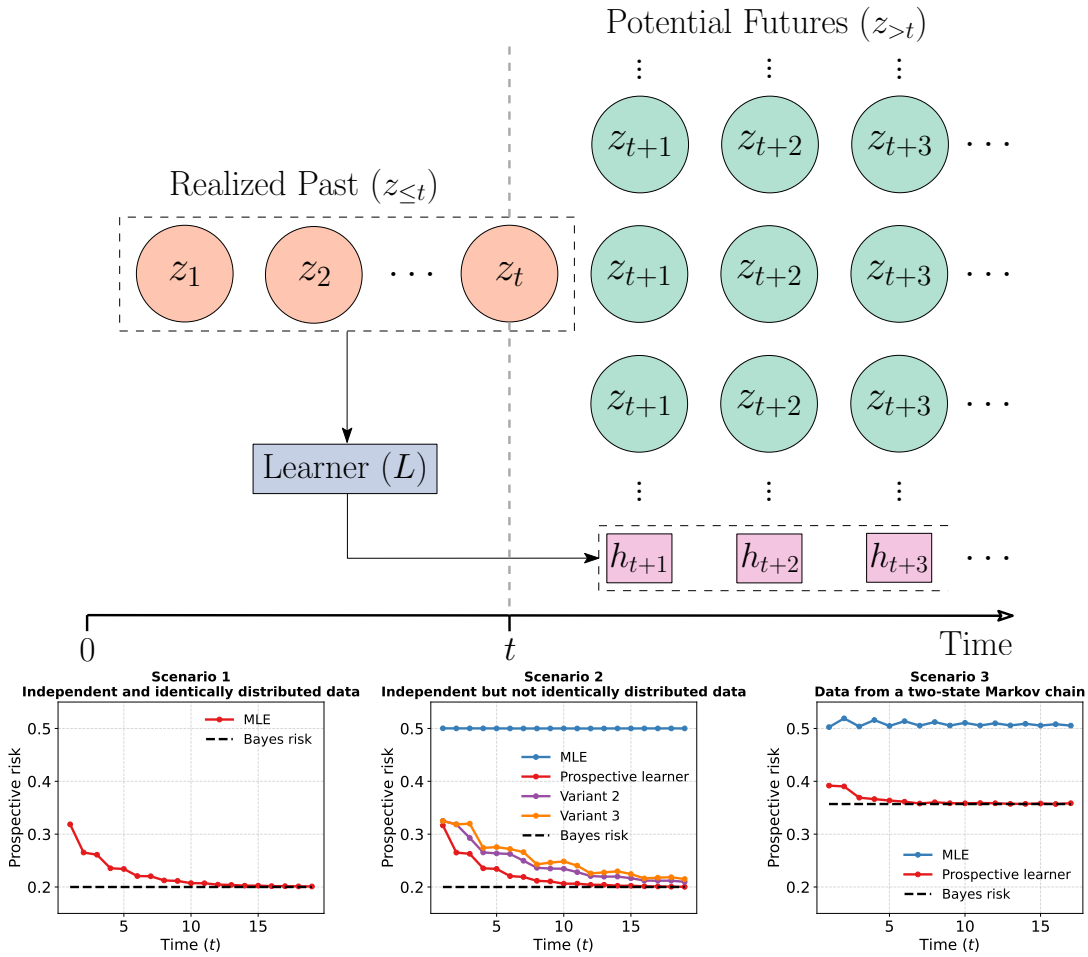


Figure 2.13: A schematic for prospective learning (left) and realizations of the examples for the four scenarios (top right); dots denote 1s and empty spaces denote 0s for $Y_t \in \{0, 1\}$ with $X_t = 1$ for all times t. Prospective risk of learners at different times is shown in the bottom panels and discussed in section 2.3.1.1. **scenario 1:** For Bernoulli probability $p = 0.2$, the maximum-likelihood estimator (MLE) in blue uses a time-agnostic hypothesis $h_t(X_t) = \mathbf{1}(\hat{p}_t > 0.5)$ where $\hat{p}_t = t^{-1} \sum_{s=1}^{t} y_s$, ties at $\hat{p}_t = 0.5$ are broken randomly. The risk of this learner converges to the Bayes risk. **scenario 2:** For Bernoulli probability $p = 0.2$, the MLE estimator (blue) performs at chance levels. A prospective learner (red) that alternates between two predictors at even and odd times converges to Bayes risk. Variants of this learner that use less information from the stochastic process (purple does not know that the data distributions at even and odd times are tied, green does not know that the distribution shifts at every time-step) also converge to Bayes risk, but more slowly. **scenario 3:** For $\theta = 0.1$ and $\gamma = 0.9$ in the discounted prospective risk, the MLE estimator (blue) again performs at chance levels. A prospective learner that computes an estimate of the transition probability of the two-state Markov chain to estimate $\mathbb{P}(Y_{t'} \mid y_t)$ for future times $t' > t$ converges to Bayes risk.

Data is independent and identically distributed. Formally, this consists of stochastic processes where $\mathbb{P}_{Z_{t'} \mid Z_{\leq t}} = \mathbb{P}_{Z_t}$ for all $t, t' \in \mathbb{N}$. As an example, consider $Y_t \sim \text{Bernoulli}(p)$ for some unknown parameter $p \in [0, 1]$. Prospective Bayes risk is equal to $\min(p, 1 - p)$ in this case. A time-agnostic hypothesis, for

example one that thresholds the maximum likelihood estimator (MLE) of the Bernoulli probability, converges to the limiting prospective Bayes risk.

Data is independent but not identically distributed. This consists of stochastic processes where $\mathbb{P}_{Z_t \mid Z_{\leq t}} = \mathbb{P}_{Z_t}$ for all $t \in \mathbb{N}$. Consider $Y_t \sim \text{Bernoulli}(p)$ if t is odd, and $Y_t \sim \text{Bernoulli}(1-p)$ if t is even, i.e., data is drawn from two different distributions at alternate times. Prospective Bayes risk is again equal to $\min(1-p, p)$ in this case. A time-agnostic hypothesis can only perform at chance level. But a prospective learner, for example one that selects a hypothesis that alternates between two predictors at even and odd times, can converge to prospective Bayes optimal risk. We can also construct variants, e.g., when the relationship between the Bernoulli probabilities are not known (Variant 1 in fig. 2.13), or when the learner does not know that the data distribution changes at every time step (Variant 2 in fig. 2.13 where we implemented a generalized likelihood ratio test to determine whether the distribution changes). The risk of these variants also converges to prospective Bayes risk, but they need more samples because they use more generic models of the stochastic process. This scenario is closely related to (online) multitask/meta-learning (Finn et al., 2017).

Data is neither independent nor identically distributed. Formally, this scenario consists of general stochastic processes. As an example, consider a Markov process $\mathbb{P}(Y_{t+1} = k \mid Y_t = k) = \theta$ with two states $k \in \{0, 1\}$ and $Y_1 \sim \text{Bernoulli}(\theta)$. The invariant distribution of this Markov process is $\mathbb{P}(0) = \mathbb{P}(1) = 1/2$. Prospective Bayes risk is also equal to 1/2. For stochastic processes that have a invariant distribution, it is impossible to predict the next state infinitely far into the future and therefore it is impossible to prospect. The prospective Bayes risk is trivially chance levels. In such situations, the learner could consider losses that are discounted over time. For example, one could use a slightly different loss than the one in eq. (2.8) to write

$$\bar{\ell}_t(h, Z) = (1-\gamma) \sum_{s=t+1}^{\infty} \gamma^{s-t-1} \ell(h_s(X_s), Y_s) \tag{2.11}$$

for some $\gamma \in [0, 1)$. In this example, we can calculate the prospective Bayes risk analytically; see section A.1.1. For $\gamma = 0.9$, $\theta = 0.1$ and the zero-one loss, limiting prospective Bayes risk is 0.357. Now consider a learner which computes the MLE of the transition matrix $\Gamma_t^{t'-t}$. It calculates $\mathbb{P}(Y_{t'} \mid y_t) = \hat{p}_{t'}$ where $[1-\hat{p}_{t'}, \hat{p}_{t'}] = \Gamma_t^{t'-t}[1-y_t, y_t]^\top$ and uses the hypothesis $h_{t'}(X_{t'}) = \mathbf{1}(\hat{p}_{t'} > 0.5)$ (ties broken ran-

domly). We can see in fig. 2.13 that this learner converges to the prospective Bayes risk. This example shows that if we model the changes in the data, then we can perform prospective learning. This scenario is closely related to certain continual learning problems (Vogelstein et al., 2020b; Ramesh and Chaudhari, 2022).

2.3.2 Theoretical foundations of prospective learning

Definition 2 (**Strong Prospective Learnability**). *A family of stochastic processes is strongly prospectively learnable, if there exists a learner with the following property: there exists a time $t'(\epsilon, \delta)$ such that for any $\epsilon, d > 0$ and for any stochastic process Z from this family, the learner outputs a hypothesis h such that $\mathbb{P}\left[R_t(h) - R_t^* < \epsilon\right] \geq 1 - \delta$, for any $t > t'$.*

This definition is similar to the definition of strong learnability in PAC learning with one key difference. Prospective Bayes risk R_t^* depends upon the realization of the stochastic process $z_{\leq t}$ up to time t. In PAC learning, it would only depend upon the true distribution of the data. Not all families of stochastic processes are strongly prospectively learnable. We therefore also define weak learnability with respect to a "chance" learner that predicts $\mathbb{E}[Y]$ and achieves a prospective risk R_t^0. [4]

Definition 3 (**Weak Prospective Learnability**). *A family of stochastic processes is weakly prospectively learnable, if there exists a learner with the following property: there exists an $\epsilon > 0$ such that for any $d > 0$, there exists a time $t'(\epsilon, \delta)$ such that for any stochastic process Z from this family, $\mathbb{P}\left[R_t^0 - R_t(h) > \epsilon\right] \geq 1 - \delta$, for any $t > t'$.*

In PAC learning for binary classification, strong and weak learnability are equivalent (Schapire, 1990) in the distribution agnostic setting, i.e., when strong and weak learnability is defined as the ability of a learner to learn any data distribution. But even in PAC learning, if there are restrictions on the data distribution, strong and weak learnability are not equivalent (Kearns, 1988). This motivates proposition 1 below. Before that, we define a time-agnostic empirical risk minimization (ERM)-based learner. In PAC learning, ERM selects a hypothesis that minimizes the empirical loss on the training data. It outputs a time-agnostic hypothesis, i.e., using data, say, $z_{\leq t}$ standard ERM returns the same predictor for future data from any time $t' > t$. There is

[4] We can also define weak learnability with respect to the prospective risk of a particular learner, even one that is not prospective. This may be useful to characterize learning for stochastic processes which do not admit strong learnability.

a natural application of ERM to prospective learning problems, defined below.

Definition 4 (**Time-agnostic ERM**). *Let $\mathcal{H}$ be a hypothesis class that consists of time-agnostic predictors, i.e., $h_t = h_{t'}$ for any $t, t' \in \mathbb{N}$ for all predictors $h \in \mathcal{H}$. Given data $z_{\leq t}$, a learner that returns*

$$\hat{h} = \underset{h \in \mathcal{H}}{argmin} \frac{1}{t} \sum_{s=1}^{t} \ell(s, h_s(x_s), y_s) \tag{2.12}$$

is called a time-agnostic empirical risk minimization (ERM)-based learner.

Time-agnostic ERM in prospective learning may use a time-dependent loss $\ell(s, h_s(x_s), y_s)$. This ERM is not very different from standard ERM in PAC learning (when instantiated with the hypothesis class that consists of sequences of predictors, that we are interested here). If data is IID (scenario 1), then there is no information provided by time in the training samples. But if there are temporal patterns in the data, like in the examples for scenarios 2 or 3, then time-agnostic ERM as defined here will return predictors that are different from those of standard ERM that uses a time-invariant loss.

Proposition 1. *There exist stochastic processes for which time-agnostic ERM is not a weak prospective learner. There also exist stochastic processes for which time-agnostic ERM is a weak prospective learner but not a strong one.*

See section A.1.2 for the proof. We do not know yet whether (or when) strong and weak learnability are equivalent for prospective learning.

2.3.2.1 Prospective empirical risk minimization

In PAC learning, the hypothesis returned by ERM using the training data can predict arbitrarily well (approximate the Bayes risk arbitrarily well with arbitrarily high probability), with a sufficiently large sample size. This statement holds if (a) there exists a hypothesis in the hypothesis class whose risk matches the Bayes risk asymptotically, and (b) if risk on training data converges to that on the test data sufficiently quickly and uniformly over the hypothesis class (Blumer et al., 1989; Alon et al., 1997). Theorem 9 is an analogous result for prospective learning.

Theorem 9 (**Prospective ERM is a strong prospective learner**). *Consider a finite family of stochastic*

processes $\mathcal{Z}$. If we have (a) consistency, i.e., there exists a hypothesis class $\mathcal{H} \subseteq (\mathcal{Y}^{\mathcal{X}})^{\mathbb{N}}$ such that $\forall Z \in \mathcal{Z}$,

$$\lim_{t\to\infty} \mathbb{E}\left[\inf_{h\in\mathcal{H}} R_t(h) - R_t^*\right] = 0, \tag{2.13}$$

where $h \in \mathcal{H}$ is a random variable in $\sigma(Z_{\leq t})$, and (b) uniform concentration of the limsup, i.e., $\forall Z \in \mathcal{Z}$,

$$\mathbb{E}\left[\max_{h\in\mathcal{H}} \left|\bar{\ell}_t(h, Z) - \frac{1}{m}\sum_{s=1}^{m} \ell(s, h_s(x_s), y_s)\right|\right] \leq \gamma_t, \tag{2.14}$$

for some $\gamma_t \to 0$ (all uniform over the family of stochastic processes), then a learner that returns

$$\hat{h} = \underset{h\in\mathcal{H}}{argmin} \frac{1}{m}\sum_{s=1}^{m} \ell(s, h_s(x_s), y_s), \tag{2.15}$$

is a strong prospective learner for this family. We define prospective ERM as the learner that uses train data $z_{\leq t}$ to implement eq. (2.15).

Proof. We would like to bound the gap between the empirical prospective risk and the true prospective risk. We define $e_t(\hat{h}) = \frac{1}{t}\sum_{s=1}^{t} l(s, h_s(x_s), y_s)$ and assuming $\mathcal{H}$ is compact, let $h_t^* = \text{argmin}_{h\in\mathcal{H}} R_t(h)$. We express the difference between the empirical and true risk as follows:

$$R_t(\hat{h}) - R_t^* = (R_t(\hat{h}) - e_t(\hat{h})) + (e_t(\hat{h}) - e_t(h_t^*)) + (e_t(h_t^*) - R_t(h_t^*)) + (R_t(h_t^*) - R_t^*).$$

Since $\hat{h}$ minimizes the empirical risk, $e_t(\hat{h}) \leq e_t(h^*)$, which results in

$$R_t(\hat{h}) - R_t^* = (R_t(\hat{h}) - e_t(h_t^*)) + (e_t(h_t^*) - R_t(h_t^*)) + (R_t(h_t^*) - R_t^*)$$

We can apply Markov's inequality to eq. (2.13) to deduce that $E\left[R_t(h_t^*) - R_t^*\right] \to 0$, which implies that $R_t(h_t^*) \xrightarrow{p} R_t^*$. As a second step, we would like to prove that $|R_t(h) - e_m(h)| \to 0\ \forall h$. Using the assumption

in eq. (2.14),

$$\begin{aligned}\mathbb{E}_{Z_{\leq t}}\left[R_t(h) - e_m(h)\right] &= \mathbb{E}_{Z_{\leq t}}\left[\mathbb{E}\left[\bar{l}(t, Z) - e_m(h) \,|\, Z_{\leq t}\right]\right] \\ &= \mathbb{E}_Z\left[\bar{l}(t, Z) - e_m(h) \,|\, Z_{\leq t}\right] \leq \gamma_t\end{aligned}$$

Since $\gamma_t \to 0$, we can use Markov's inequality to show that

$$P\left(|R_t(h) - e_t(h)| \geq \epsilon\right) \leq \frac{1}{\epsilon}\mathbb{E}\left[\bar{l}(t, Z) - e_t(h)\right] \leq \frac{\gamma_t}{\epsilon}$$

Hence, $R_t(h) \xrightarrow{p} e_t(h)$ for all h. Using the results from both steps, we get, $R_t(\hat{h}) - R_t^* \to 0$, which implies that prospective ERM is a strong prospective learner. □

In this version of the theorem, we assume that the lim is always well-defined. A more general version of this theorem would use the lim sup instead. The first condition, eq. (2.13), is analogous to the consistency condition in PAC learning. In simpler words, it states that the Bayes risk can be approximated well using the chosen sequence of hypothesis classes $\{\mathcal{H}_t\}_{t=1}^{\infty}$. The second condition, eq. (2.14), is analogous to concentration of measure in PAC learning, it requires that the lim in eq. (2.8) is close to an empirical estimate of the lim (the second term inside the absolute value in eq. (2.14)).

At each time t, prospective ERM in eq. (2.15) selects the best hypothesis $\hat{h} \in \mathcal{H}$ for future times $t' > t$, that minimizes an empirical estimate of the limsup using the training data $z_{\leq t}$. Prospective ERM can exploit the difference between the latest datum in the training set with time t and the time for which it makes predictions t' by selecting specific sequences inside the hypothesis class $\mathcal{H}$. For example, in scenario 2 it can select sequences where alternating elements can be used to predict on data from even and odd times.

Remark 10 (How to implement prospective ERM?). *An implementation of prospective ERM is therefore not much different than an implementation of standard ERM, except that there are two inputs: time s and the datum x_s. Suppose we use a hypothesis class where each predictor is a neural network, this could be a multi-layer perceptron or a convolutional neural network. The training set $z_{\leq t}$ consists of inputs x_s along with corresponding time instants s and outputs y_s. To implement prospective ERM, we modify the network to take (s, x_s) as input (using any encoding of time, we discuss one in section 2.3.3) and train the network*

to predict the label y_s. At inference time, this network is given the input $(t', x_{t'})$ to obtain the prediction $y_{t'}$. Note that if prospective ERM is implemented in this fashion, the learner need not explicitly calculate the infinite sequence of predictors.

Corollary 2.3.1. *There exist stochastic processes for which time-agnostic ERM is not a strong prospective learner, but prospective ERM is a strong learner.*

In PAC learning, one first proves uniform convergence for a finite hypothesis class. This can then be used to calculate the sample complexity of ERM, or extended to infinite hypothesis classes using constructions such as VC-dimension and covering numbers (Vapnik and Chervonenkis, 2015). Although we do not have such results for prospective learning for a general family of stochastic process, for periodic processes we can calculate the sample complexity.

2.3.2.2 Prospective ERM on periodic processes

Suppose we have a stochastic process such that $Z_t \sim P_{(t \bmod T)}$ for some known period T, i.e., data is independent across time but not identically distributed, and the loss function $\ell(t, \cdot, \cdot)$ is time-invariant. Scenario 2 is a special case with $T = 2$. Assume that we can find a hypothesis class $\mathcal{G}$ that contains the Bayes estimator for each P_t with $t \in \{1, \ldots, T\}$. Then $\mathcal{H}_T = \{h : h_{t+T} = h_t \text{ and } h_t \in \mathcal{G}\ \forall t\}$. Note that even if we do not know the period, we can still implement prospective ERM using the hypothesis class $\cup_{t\in\{1,\cdots T\}}\mathcal{H}_t$. Prospective ERM is therefore a strong prospective learner if the period T is bounded.

Remark 11 (**Implementing prospective ERM for periodic processes**). *If $\mathcal{G}$ has a finite VC-dimension, choosing $\mathcal{H} = \mathcal{H}_T$ as the hypothesis class guarantees that $\lim_{m\to\infty} \frac{1}{m} \sum_{s=1}^{m} \ell(s, h_s(x_s), y_s)$ is well defined. We can therefore choose*

$$\hat{h} = \underset{h\in\mathcal{H}}{argmin} \frac{1}{t} \sum_{s=1}^{t} \ell(s, h_s(x_s), y_s)$$

in eq. (2.15). In other words, implementing prospective ERM for a periodic process boils down to solving T different time-agnostic ERM problems, each using data $\{z_{sT+k}\}_{s=0}^{\infty}$, $k \in \{1, \ldots, T\}$. Observe that this is identical to the prospective learner we used for the example in scenario 2 and fig. 2.13.

Remark 12 (**Sample complexity of prospective ERM for a periodic process**). *We can calculate the sample complexity by exploiting the relatedness of the different distributions in the periodic process. First assume $t > T$, i.e., at least one sample from each distribution is available. We again pick $\mathcal{H} = \mathcal{H}_T$. Let us assume*

that $\hat{h}_t \in \mathcal{G}$ *for all times* t. *Let* $C \equiv C(\epsilon/16, \mathcal{G}^T)$ *denote the covering number of a hypothesis class of* T*-length sequences of hypotheses* $\mathcal{G}^T = \{(h, \ldots, h) : h \in \mathcal{G}\}$ *using balls of radius* $\epsilon/16$ *with respect to loss* ℓ. *Then, using Baxter (2000, Theorem 4) we can show that, if*

$$t \geq \max\left\{\frac{64}{\epsilon^2}\log\frac{4C(\epsilon, \mathcal{G}^T)}{d}, \frac{16T}{\epsilon^2}\right\}, \tag{2.16}$$

then for prospective ERM in eq. (2.15) *we have*

$$\mathbb{E}\left[R_t(\hat{h})\right] \leq \lim_{t\to\infty}\mathbb{E}\left[\inf_{h\in\mathcal{H}} R_t(h)\right] + 2\epsilon,$$

with probability at least $1-\delta$. *The sample complexity in eq.* (2.16) *is dominated by the first term in the curly brackets; Baxter (2000, Lemma 5) shows that* $C(\epsilon, \mathcal{G}^T) \leq (C(\epsilon, \mathcal{G}))^T$. *Sample complexity of prospective ERM grows at most linearly with the period* T, *as one would expect.*

2.3.3 Experiments on prospective learning

This section demonstrates that we can implement prospective ERM on prospective learning problems constructed on synthetic data, MNIST and CIFAR-10. In practice, prospective ERM may approximately achieve the guarantees of theorem 9. We will focus on the distribution changing, independently or not (scenario 2, 3). Recall that scenario 1 is the same as the IID setting used in standard supervised learning problems. We discuss experiments that check whether large language models can do prospective learning in section 2.3.3.5.

Learners and hypothesis classes. Task-agnostic online continual learning methods are the closest algorithms in the literature that can address situations where data evolve over time. We use the following three methods.

(i) **Follow-the-Leader** minimizes the empirical risk calculated on all past data and is a no-regret algorithm (Cesa-Bianchi and Lugosi, 2006). We note that while this is a popular online learning algorithm, we do not implement the algorithm in an online fashion.

(ii) **Online SGD** fine-tunes the network using new data in an online fashion. At every time step, weights of the network are updated once using the last eight samples.

(iii) **Bayesian gradient descent** (Zeno et al., 2018) is an online continual learning algorithm designed to address situations where the identity of the task is not known during both training and testing, i.e., it implements continual learning without knowledge of task boundaries.

These three methods are not explicitly designed for prospective learning, but are designed to address the changing data distribution t.[5]

We calculate the prospective risk of the predictor returned by these methods; note that they do not output a time-varying predictor and consequently, these methods output a time-agnostic hypothesis. As a result, when we evaluate the prospective risk of these methods, we use the same hypothesis for all future time. For all three methods, we use a multi-layer perceptron (MLP) for synthetic data and MNIST, and a convolutional neural network (CNN) for CIFAR-10.

For **prospective ERM** the sequence of predictors is built by incorporating time as an additional input to an MLP or CNN as follows. For frequencies $\omega_i = \pi/i$ for $i = 1, \ldots, d/2$, we obtain a d-dimensional embedding of time t as $\varphi(t) = (\sin(\omega_1 t), \ldots, \sin(\omega_{d/2} t), \cos(\omega_1 t), \ldots, \cos(\omega_{d/2} t))$. This is similar to the position encoding in Vaswani et al. (2017). A predictor $h_t(\cdot)$ uses a neural network that takes as input, an embedding of time $\varphi(s)$, and the input x_s to predict the output y_s for any time $s \in \mathbb{N}$. Using such a time embedding is useful in prospective learning because one does not need to explicitly maintain the infinite sequence of predictors $h \equiv (h_1, \ldots,)$.

Training setup. We use the zero-one error $\mathbf{1}\{\hat{y} \neq y\}$ to calculate prospective risk for all problems; all learners are trained using a standard surrogate of this objective, the cross-entropy loss. For all experiments, for each time t, we calculate the prospective risk $R_t(h)$ in eq. (2.9) of the hypothesis created by these learners for a particular realization of the stochastic process $z_{\leq t}$. For each prospective learning problem, we generate a sequence of 50,000 samples. Learners are trained on data from the first t time steps ($z_{\leq t}$) and prospective risk is computed using samples from the remaining time steps. Except for online SGD and Bayesian gradient

[5] There are many algorithms in the existing literature that the reader may think of as reasonable baselines. We have chosen a representative and relevant set here, rather than an exhaustive one. For example, online meta-learning approaches are close to online-SGD; since the learner fine-tunes on the most recent data. Algorithms in the literature on time-series (i) focus on predicting future data, say, $Y_{t'}$ given past data $y_{\leq t}$ without taking covariates $X_{t'}$ or some exogenous variables $X_{\leq t}$ into account, (ii) can usually only make predictions for a pre-specified future context window (Lim et al., 2021), and (iii) work for low-dimensional signals (unlike images).

descent, learners corresponding to different times are trained completely independently. See section A.6.3 for more details.

Remark 13 (Why we do not use existing benchmark continual learning scenarios). *The tasks constructed below resemble continual learning benchmark scenarios such as Split-MNIST or Split-CIFAR10 (Zenke et al., 2017) where data from different distributions are shown sequentially to the learner. However, there are three major differences. First, in these existing benchmark scenarios, data distributions do not evolve in a predictable fashion, and prospective learning would not be meaningful. Second, existing scenarios consider a fixed time horizon. We are keen on calculating the prospective risk for much longer horizons whereby the differences between different learners are easier to discern; our experiments go for as large as 30,000 time steps. Third, our tasks have the property that the Bayes optimal predictor changes over time.*

2.3.3.1 Prospective learners for independent but not identically distributed data (scenario 2)

We create tasks using synthetic data, MNIST and CIFAR-10 datasets to design prospective learning problems when data are independent but not identically distributed across time (scenario 2)

Dataset and Tasks. For the synthetic data, we consider two binary classification problems ("tasks") where the input is one-dimensional. Inputs for both tasks are drawn from a uniform distribution on the set $[-2,-1] \cup [1,2]$. Ground-truth labels correspond to the sign of the input for Task 1, and the negative of the sign of the input for Task 2. For MNIST and CIFAR-10 we consider 4 tasks corresponding to data from classes 1-5, 4-7, 6-9 and 8-10 in the original dataset, i.e., the first task considers classes 1-5 labeled 1-5 respectively, the second task considers classes 4-7 labeled 1-4, the third task considers classes 6-9 labeled 1-4 and the last task considers labels 8-10 labeled 1-3. In other words, images from class 1 in task 1, class 4 from task 2 and class 6 from task 3 are all assigned the label 1. For the prospective learning problem based on synthetic data, the task switches every 20 time steps. For MNIST and CIFAR-10, the data distribution cycles through the 4 tasks, and the distribution of data changes every 10 time-steps. For more details, see section A.6.3.

Figure 2.14 shows that **prospective ERM can learn problems in scenario 2 when the data are independent but not identically distributed**. For prospective learning problems constructed from synthetic data, the risk of prospective ERM converges to prospective Bayes risk over time. For the MNIST and CIFAR-10 prospective problems, the prospective learning risk drops precipitously. In contrast, online learning base-

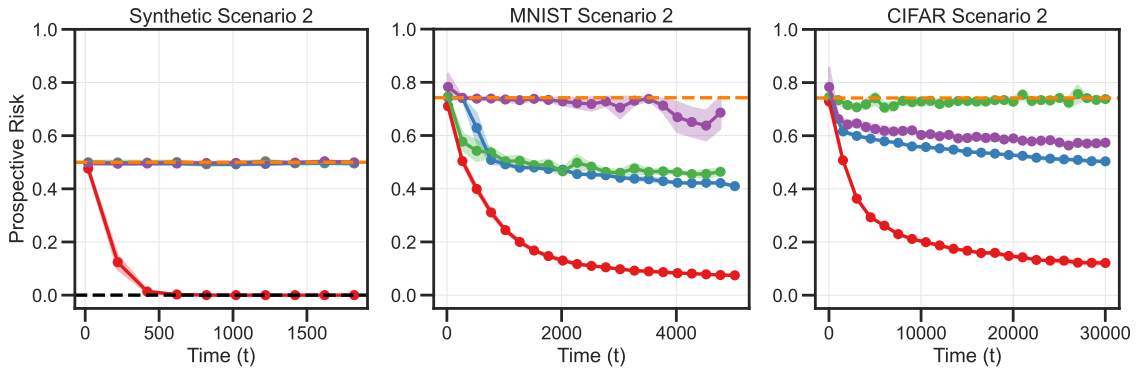


Figure 2.14: Prospective ERM can achieve good instantaneous and prospective risk in scenario 2. Left: Instantaneous and prospective risks for problems constructed using synthetic data (see text) across 5 random seeds (which govern the sequence of samples and the weight initializations of neural networks). Instantaneous risk spikes when the task switches for many online learning baseline algorithms. In contrast, prospective ERM has minimal spikes at later times and both instantaneous and prospective risks eventually converge to zero. **Right:** Prospective risk for different baseline algorithms and prospective ERM for tasks constructed using MNIST and CIFAR-10 for scenario 2. In all three cases, the risk of prospective ERM approaches Bayes risk while online learning baselines considered here do not achieve a low prospective risk. For comparison, the chance prospective risk is 0.5 for synthetic data and 0.742 for MNIST and CIFAR-10 tasks.

lines discussed above achieve a far worse prospective risk. Observe that Follow-the-Leader (blue) performs as well, or better, as online SGD and Bayesian GD. This is not surprising; while the ERM models corresponding to each time t were trained independently, the networks corresponding to online SGD and Bayesian GD were trained in an online fashion. In practice, it is often difficult to tune online learning methods effectively (Li et al., 2020a). [6]

2.3.3.2 Prospective learners when data are neither independent nor identically distributed

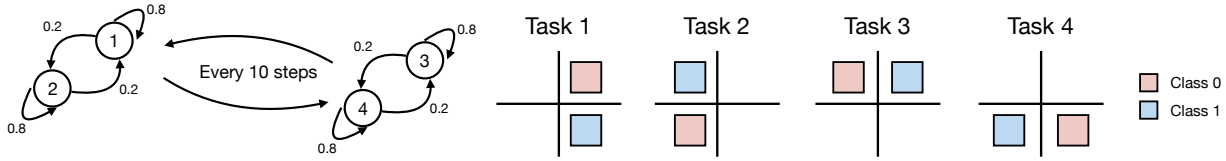


Figure 2.15: Left: For MNIST and CIFAR-10, we consider 4 tasks corresponding to the classes 1-5, 4-7, 6-9 and 8-10. Using these tasks, we construct scenario 3 problems corresponding to a stochastic process which is a hierarchical hidden Markov model. After every 10 time-steps, a different Markov chain governs transitions among tasks (one Markov chain for tasks 1 and 2, and another for tasks 3 and 4). This ensures that the stochastic process does not have a stationary distribution. **Right:** For synthetic data, the 4 tasks are created using two-dimensional input data as shown pictorially above. The four parts of the input domain are $\{(x_1, x_2) : 1 \leq x_1, x_2 \leq 2\}$, $\{(x_1, x_2) : 1 \leq x_1 \leq 2$, and $-2 \leq x_2 \leq -1\}$, $\{(x_1, x_2) : -2 \leq x_1, x_2 \leq -1\}$ and $\{(x_1, x_2) : -2 \leq x_1 \leq -1$ and $1 \leq x_2 \leq 2\}$. Colors indicate classes. The hierarchical hidden Markov model for transitions among the tasks is identical for MNIST and CIFAR-10.

[6] For CNNs on CIFAR-10, if one concatenates the time embedding directly to the input images as opposed to concatenating to a layer before softmax, like it is done here, the prospective risk in fig. 2.14 (right) is much higher (worse by almost 0.2; See fig. A.29). The implementation details of the time embedding matter when implementing prospective learners in practice, even if theorem 9 is true in general.

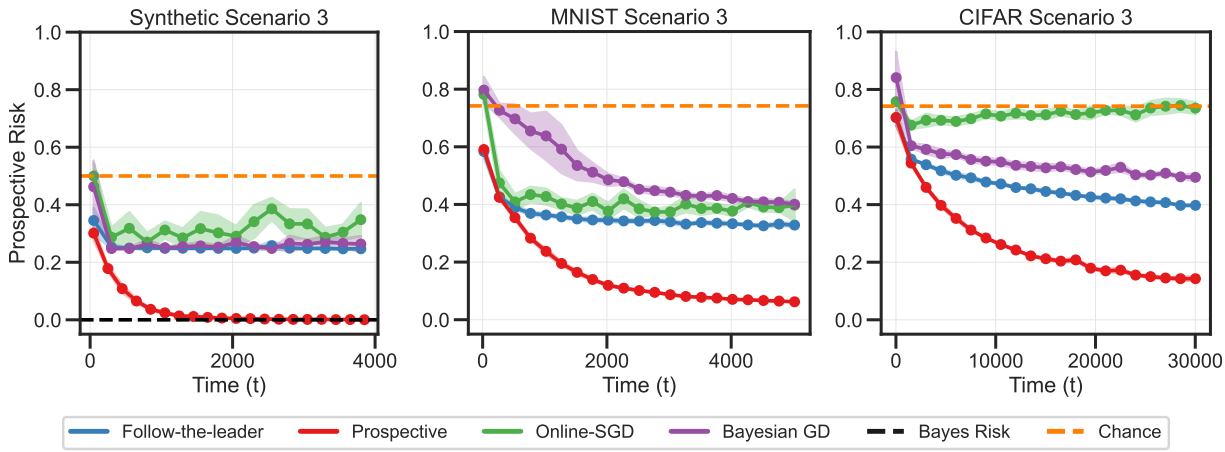


Figure 2.16: Prospective ERM can achieve good prospective risk in scenario 3. Prospective risk across 5 random seeds (which govern the sequence of samples and the weight initializations of neural networks). In all three cases, the risk of prospective ERM approaches Bayes risk while a number of baseline algorithms do not achieve a low prospective risk. Stochastic processes in these problems corresponding to scenario 3 do not have an invariant distribution. This is why a time-agnostic hypothesis (ERM) that is constructed by the baseline algorithms does not achieve a good prospective risk.

Dataset and Tasks. For synthetic data, we construct 4 binary classification problems with two-dimensional input data (see fig. 2.15 and caption for details). For CIFAR-10 and MNIST, we consider four tasks corresponding to the classes 1-5, 4-7, 6-9 and 8-10. Using these tasks, we construct problems where the data distribution is governed by a stochastic process which is a hierarchical hidden Markov model (scenario 3). After every 10 time-steps, a different Markov chain governs transitions among tasks (one Markov chain for tasks 1 and 2, and another for tasks 3 and 4, as shown in fig. 2.15). These choices ensure that the stochastic process does not have a stationary distribution.[7]

As fig. 2.16 shows, **prospective ERM can prospectively learn problems when data is both independent and not identically distributed (scenario 3)**. Stochastic processes in these problems corresponding to scenario 3 do not have a stationary distribution. This is why a time-agnostic hypothesis (Follow-the-Leader) does not achieve a good prospective risk, unlike prospective ERM.

2.3.3.3 Markov chain with periodic resets

Dataset and Tasks. For synthetic data, we consider the 2 binary classification problems described in section 2.3.3.1. For CIFAR-10 and MNIST, we consider 2 tasks corresponding to the classes 1-5, and the classes

[7] As we discussed in scenario 3, prospective Bayes risk can be trivial in situations when the stochastic process has a stationary distribution.

1-5 but with each class y relabeled to $(y+1) \pmod 5$. Using these tasks, we construct Scenario 3 problems corresponding to a stochastic process which is a hidden Markov model on two states. The tasks are governed by a Markov process with transition matrix $P(S_{t+1} = k \mid S_t = k) = 0.1$, where S_t is the task at time t. Additionally after every 10 time-steps, the state of the Markov chain is reset to the first task. This ensures that the stochastic process does not have a stationary distribution. Similar to the previous experiments, for each problem, we generate a sequence of 50,000 samples. Learners are trained on data from the first t time steps ($z_{\leq t}$) and prospective risk is computed using samples from the remaining time steps.

Learners and hypothesis classes. For this scenario, we conduct experiments using follow-the-leader and prospective ERM. Both methods use MLPs for synthetic and MNIST tasks, and a CNN for the CIFAR-10 task. Note that prospective ERM uses an embedding of time as input in addition to the datum. Training and evaluation setup is identical to that of scenario 2.

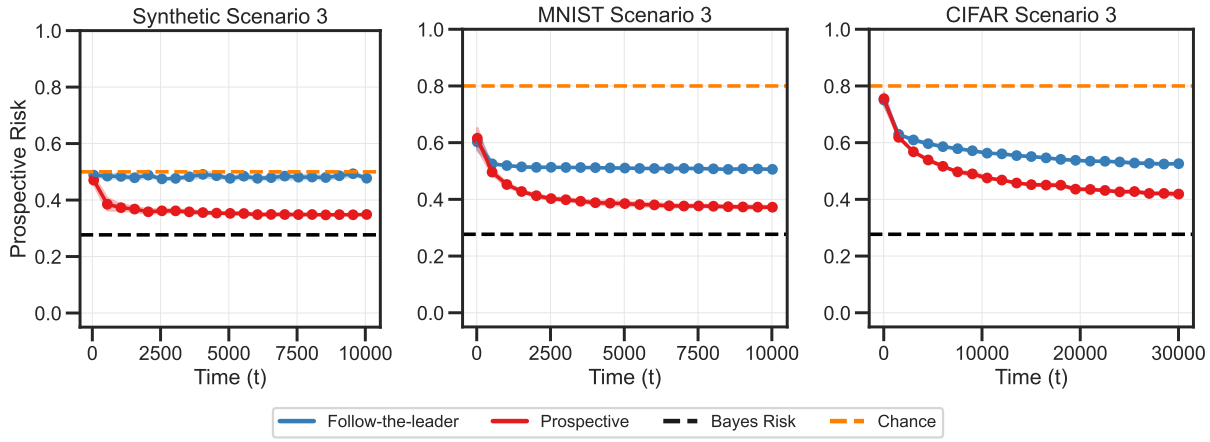


Figure 2.17: Prospective ERM can achieve good prospective risk in another instance of scenario 3 We plot the prospective risk across 5 random seeds (which govern the sequence of samples and the weight initialization of the neural networks). In all three cases, the risk of prospective ERM approaches Bayes risk while Follow-the-Leader does not achieve a low prospective risk. Bayes risk for MNIST and CIFAR-10 problems is calculated by assuming that Bayes risk on individual tasks is zero.

As Figure 2.17 shows, **prospective ERM can also prospectively learn another instance of a problem in Scenario 3 when data is neither independent nor identically distributed**.

2.3.3.4 Stationary Markov chain

Dataset and Tasks. For synthetic data, we consider the 2 binary classification problems described in section 2.3.3.1. For CIFAR-10 and MNIST, we consider 2 tasks corresponding to the classes 1-5, and the classes 1-5 but with each class y relabeled to $(y+1) \mod 5$. Using these tasks, we construct Scenario 3 problems

corresponding to a stochastic process which is a hidden Markov model on 2 states. The tasks are governed by a Markov process with transition matrix $P(S_{t+1} = k \mid S_t = k) = 0.1$, where S_t is the task at time t. Unlike the previous subsection (Figure 2.17), in this experiment, the Markov chain equilibriates to the stationary distribution. Similar to the previous experiments, for each problem, we generate a sequence of 50,000 samples. Learners are trained on data from the first t time steps ($z_{\leq t}$) and prospective risk is computed using samples from the remaining time steps.

Learners and hypothesis classes. For this scenario, we conduct experiments using follow-the-leader and prospective ERM. Both methods use MLPs for the synthetic and MNIST tasks, and a CNN for the CIFAR-10 task. Note that prospective ERM uses an embedding of time as input in addition to the datum. Training is identical to that of scenario 2. For evaluation, we compute the empirical prospective risk in fig. 2.18 and empirical discounted prospective risk in fig. 2.19.

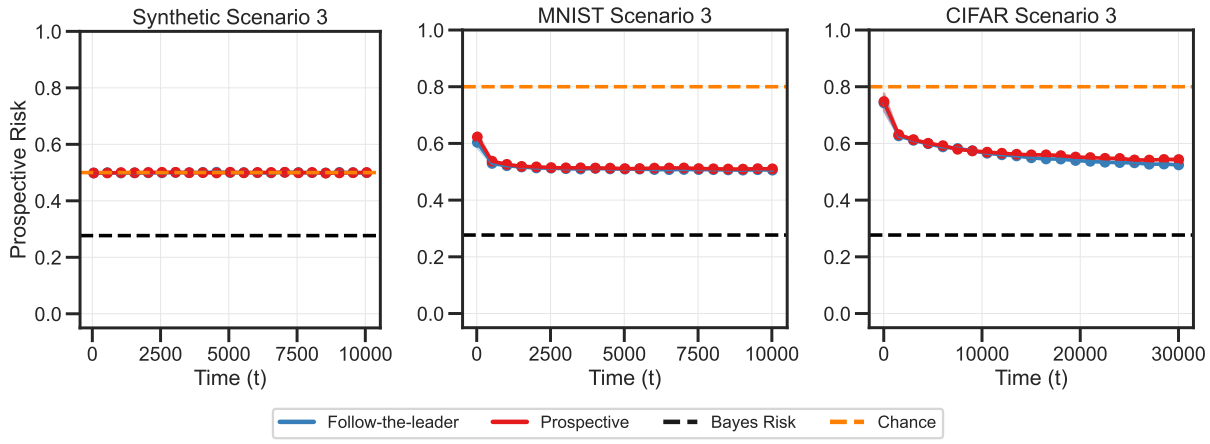


Figure 2.18: For a task defined on a stationary Markov process, the Bayes risk is trivial and can be achieved by a hypothesis that doesn't change over time. We plot the prospective risk across 5 random seeds (which govern the sequence of samples and the weight initialization of the neural networks). In all three cases, both follow-the-leader and prospective ERM approach the Bayes risk. The stationary distribution has an equal probability of seeing either task and a fixed hypothesis can achieve Bayes risk on this problem.

2.3.3.5 Large language models may not be good prospective learners

It is an interesting question whether LLMs which are trained using auto-regressive likelihoods with Transformer architectures can do prospective learning. To study this, we used LLama-7B (Touvron et al., 2023) and Gemma-7B (Team et al., 2024) to evaluate the prospective risk for scenarios 1, 2, 3. The prompt contains a few samples from the stochastic process (sub-sequences of (Y_t) consisting of 0s and 1s) and an English language description of the family of stochastic processes that generated the samples. The LLM is tasked with

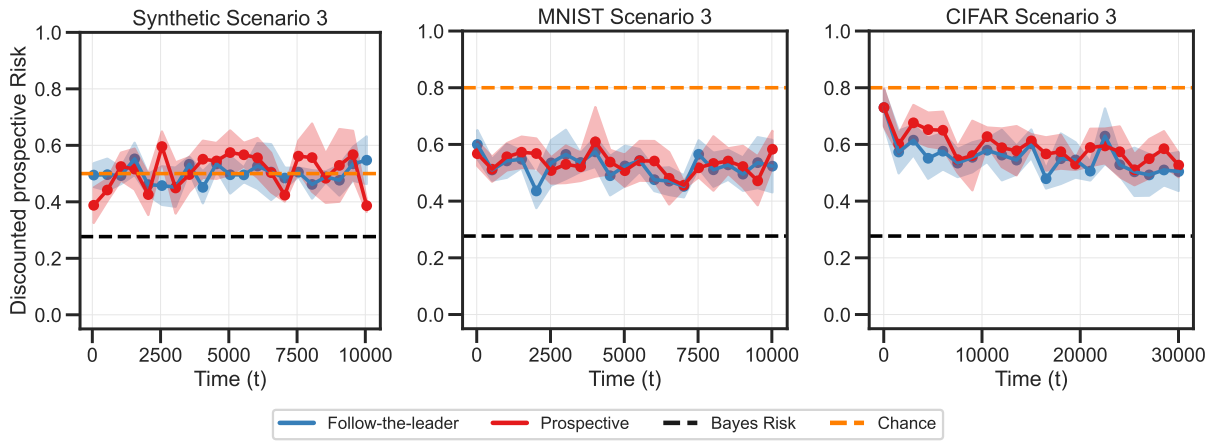


Figure 2.19: Both prospective ERM and follow-the-leader achieve similar discounted prospective risks (with discount factor 0.95). We plot the discounted prospective risk across 5 random seeds. Both follow-the-leader and prospective ERM achieve similar discounted risks. Note that the error bars are larger since the risk is computed over fewer samples, i.e., the discount factor reduces the effective number of test data points.

completing the prompt with the next 20 most likely sequence of samples.

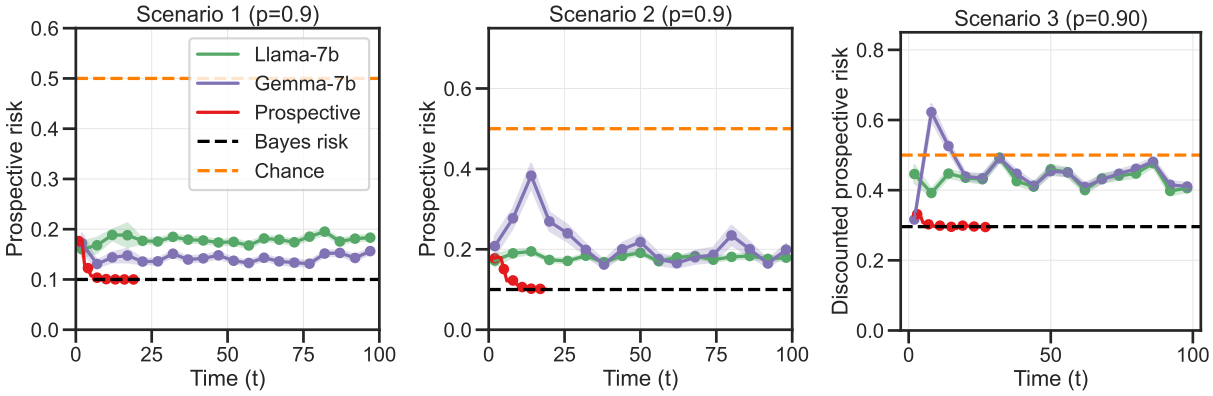


Figure 2.20: The prospective risk of LLMs when evaluated on the three scenarios, when averaged over draws of the training data. The LLM does not improve with more data unlike a prospective MLE-learner. This suggests that LLMs are incapable of prospection.

Selecting the appropriate prompt LLMs can be brittle and are known to generate different completions depending on if the prompt was in English, Thai or Swahili (Deng et al., 2023). This makes it difficult to evaluate prospective learning in LLMs. In our experiments, we do not describe prospective risk or other details about prospective learnability in the prompt. We simply describe the data generating process and some samples from this process in the prompt and the model generates the most likely completion. The prompts are described in detail in section A.6.3.3; we also experimented with a few variants of these prompts.

We use greedy decoding to generate a sequence of tokens, i.e., the token with the highest probability is

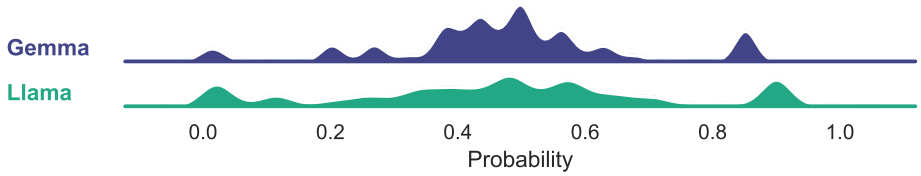


Figure 2.21: We prompt LLMs to generate the outcomes of 10 Bernoulli trials with $p = 0.75$. We plot the probability of generating token 1 over all possible sequences of 10 Bernoulli trials and find that the outcomes are generated with probabilities that range from 0 to 1 with an average of 0.5. Ideally, the token 1 should should always be generated with $p = 0.75$, i.e., the LLMs cannot simulate outcomes of a Bernoulli distribution.

sampled at every step. We vary the number of time-steps in the prompt from 1 to 100. For a particular value of time t, we generate 20 more tokens and compute (an estimate of) the prospective risk of this completion; this is the test data. We report the prospective risk computed on 100 different realizations of the stochastic process, i.e., each point in fig. 2.20 is the prospective risk on the next 20 samples, averaged over 100 realizations of the training data. In fig. 2.20, we find that LLMs do not obtain better prospective risk with more samples, i.e., Llama-7B and Gemma-7B do not seem to be doing prospective learning. It is quite surprising that they do not achieve Bayes risk even on independent and identically distributed data. We note that these experiments do not definitively answer whether LLMs can learn prospectively.

Can LLMs even generate outcomes of a sequence of Bernoulli trials? We prompted an LLM to generate a sequence of 0s and 1s sampled from a Bernoulli distribution with probability $p = 0.75$. We then plot the probability of generating each 0 or 1, for all sequences of length 10 in fig. 2.21. Ideally, the strip plot would be concentrated around 0.25 for 0 and 0.75 for 1, i.e., 0s should be generated with frequency close to 0.25. However, we find this is not the case and LLMs seem incapable of even generating a sequence of Bernoulli trials. This provides some context to the results discussed above. LLMs do not seem to be doing prospective learning, but they cannot even sample from a Bernoulli distribution under these experimental conditions.[8]

[8] Responses of ChatGPT-turbo and GPT-4o were more verbose compared to those of Llama-7B and Gemma-7B. ChatGPT responded correctly to scenario 1, perhaps as a result of using a scratchpad (Nye et al., 2021; Wei et al., 2022b) for generating the results of intermediate steps of the algorithm. But it did not achieve a small prospective risk for scenarios 2 and 3. Gemini and GPT-4o refused to give a complete response to scenario 3 and only outlined the sequence of steps, albeit correctly.

2.4 Related Work

Theoretical work on learning from multiple tasks Works such as Baxter (2000); Maurer (2006), or recent ones like Du et al. (2020); Tripuraneni et al. (2020b) study a shared feature generator with task-specific classifiers, and show that the sample complexity of learning a task improves if true task-specific classifiers are diverse enough. It is also appreciated that such a shared feature generator may not exist for dissimilar tasks. A different perspective on the problem can be found in Crammer et al. (2008); Ben-David et al. (2010); Ben-David and Borbely (2008) who show that learning diverse tasks requires a larger feature generator and, thereby, more samples. Model Zoo builds upon Hanneke and Kpotufe (2019, 2020) to construct the transfer exponent in section 2.1.2; Their work shows that even in very favorable settings, for example, when all tasks have the same optimal classifier, having access to a large number of tasks may not help. Model Zoo is strongly influenced from these results and we think of it as essentially a way to circumvent them.

There are a number of algorithmic tools to estimate task relatedness, e.g. (Kumar and Daume III, 2012; Evgeniou et al., 2005; Cavallanti et al., 2010), and although such methods are popular in areas such as transfer learning (Pentina and Lampert, 2015; Jaakkola and Haussler, 1999), one cannot apply them in continual learning because we do not know the tasks beforehand. As section 2.1.2 shows, task relatedness is critical for good learning. So, taking inspiration from AdaBoost (Schapire and Freund, 2013), Model Zoo uses a simple indicator of which past tasks can benefit from future ones, these are the ones with low accuracy under the current ensemble.

Approaches for continual learning. Catastrophic forgetting has been the focus of a number of continual learning techniques, e.g., episodic memory-based ones (Lopez-Paz and Ranzato, 2017; Chaudhry et al., 2019a; Farajtabar et al., 2020; Guo et al., 2020), data replay (Robins, 1995; Shin et al., 2017; Lee et al., 2017), new architectures (Serra et al., 2018), generative replay-based (Mocanu et al., 2016; Shin et al., 2017; Liu et al., 2020; Ven et al., 2020), ensemble-based (Aljundi et al., 2017; Wen et al., 2020) and methods that select locally redundant directions in the weight space (Kirkpatrick et al., 2017; Aljundi et al., 2018; Mallya et al., 2018; Zenke et al., 2017; Chaudhry et al., 2018). Variational methods, for example, (Nguyen et al., 2018; Farquhar and Gal, 2019a), sequentially update a posterior over the weights and have an elegant foundation in Bayesian methods, but implementing them for large datasets remains a chal-

lenge. Despite intense activity in this area, an effective solution to forgetting remains largely unknown.

Model Zoo embraces the fact that forgetting is a fundamental phenomenon of learning multiple tasks and therefore splitting the capacity may be essential; our results indicate that this approach is effectively at tackling forgetting. This approach also significantly improves other key metrics, for example forward-backward transfer and computational complexity of training and inference that have received limited attention (Díaz-Rodríguez et al., 2018). Let us note that Model Zoo is designed for the task-incremental continual learning setting (Van de Ven et al., 2022).

A single shared feature generator (i.e., hard parameter sharing) is a popular architecture (Kirkpatrick et al., 2017; Lopez-Paz and Ranzato, 2017; Rebuffi et al., 2017; Nguyen et al., 2018; Mirzadeh et al., 2020b; Chaudhry et al., 2019b). It has been recognized that this is not sufficient; this has resulted in methods for soft-parameter sharing that either design or learn specialized routing architectures (Rosenbaum et al., 2017; Sun et al., 2019; Fernando et al., 2017; Devin et al., 2017; Misra et al., 2016; Vandenhende et al., 2019). Model Zoo is a very simplistic instantiation of parameter isolation, or growing (Rusu et al., 2016; Mallya and Lazebnik, 2018; Xu and Zhu, 2018). Model Zoo trains on one episode and never updates the model again but its accuracy does play a role in determining whether a *new* model should be used for that past task, or not. To extend the analogy, just like soft-parameter sharing architectures use, say gradient conflict (Aljundi et al., 2018) or attention (Serra et al., 2018), to determine which synapses to share, Model Zoo uses the training loss of the ensemble to decide what task the new model should be trained upon.

Distribution shift. Prospective learning (De Silva et al., 2023) is equivalent to PAC learning (Vapnik, 1999) when data is IID. Situations in which this assumption may not be valid are often modeled as a distribution shift between train and test data (Quiñonero-Candela, 2009). Techniques such as propensity scoring (Agarwal et al., 2011; Fakoor et al., 2024) or domain adaptation (Daume, 2007; Ben-David et al., 2010) reweight or map train / test data to return to the IID setting; techniques such as domain invariance (Arjovsky, 2020; Blum-Smith and Villar, 2023) build a statistic that is invariant to the distribution shift. Typically, the loss is unchanged across train and test data. If the set of marginals $\{\mathbb{P}(Z_t)\}$ of the stochastic process only has two elements, then PL is equivalent to the classical distribution shift setting. But otherwise, in PL, data is correlated over time, distributions (marginals) can change multiple times.

Sequential decision making and online learning. Prospective learning builds upon works on learning from streaming data. But our goals are different. For example, Gama et al. (2013) minimize the error on samples from a stationary process; Hayes et al. (2020) minimize the error on a fixed held-out dataset or on all past data—neither of these emphasizes prospection. There is a rich body of work on sequential decision making, for example, predicting a finite-state stationary ergodic process from past data (Cover, 1975). Even in this simple case, there does not exist a consistent estimator using the finite past $Z_{1:t}$ (Bailey, 1976; Ryabko, 1988; Ornstein, 1978). This is also true for regression (Morvai et al., 1996; Nobel, 2003), when the true hypothesis f^* s.t. $Y_t = f^*(X_t)$ is fixed. In other words, Bayes risk R^*_t in theorem 9 may be nonzero in prospective learning even for finite-state stationary ergodic processes. Hanneke (2021) lifts the restriction on stationarity and ergodicity. They obtain conditions on the input process X for consistent inductive (predict at time $t' > t$ using data up to t), self-adaptive (predict at time t' using $Z_{\leq t}$ and $X_{t+1:t'}$) and online learning (predict at t' using $Z_{\leq t'}$). They prove the existence of a learning rule that is consistent for every X that admits self-adaptive learning. If X is "smooth", i.e., input marginals have a similar support over time, then the ERM has a sample complexity similar to that of the IID setting (Block et al., 2024). Haghtalab et al. (2022) give algorithmic guarantees for several online estimation problems in this setting.

The true hypothesis in prospective learning can change over time. This is different from the continual learning setting, where we can find a common hypothesis for tasks at all times (Peng et al., 2023), and this is why our proofs work quite differently from existing ones in the literature. Instead of a hypothesis class $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$, we define the notion of a hypothesis class that consists of sequences of predictors, i.e., subset of $(\mathcal{Y}^{\mathcal{X}})^{\mathbb{N}}$; we can do ERM in this new space. Instead of consistency of prediction as in Hanneke (2021), we give guarantees for strong learnability, i.e., convergence of the ERM risk to the Bayes risk.

CHAPTER 3

SELF-SUPERVISED LEARNING FOR MULTIPLE TASKS

The previous chapter was dedicated to learning representations using labeled data from multiple tasks. However, the overwhelming preference today is to pretrain representations using unlabeled data, largely because it is easier to access massive amounts of it. The unlabeled datasets are often orders of magnitude larger than the labeled data, which helps us train representations that often parallel, if not outperform, ones trained on unlabeled data.

In this chapter, we consider the problem of learning representations for multiple tasks using just unlabeled data. However, the unlabeled data is not divided up into well-defined tasks. The main contribution of this chapter is to use reference priors to develop a formal theory for self-supervised learning. In particular, we treat the problem of pretraining on unlabeled data as identical to the task of selecting a prior over the weights. Using the theory of reference priors, we argue that it is optimal to train an ensemble of models that span the space of typical tasks, as opposed to training one single model — we saw something very similar in the first chapter with Model Zoo. We end this chapter with a study on a popular objective that is used to train most self-supervised models today: masked reconstruction. We find that the scale of the noise controls the scale of the features learned from pretraining; we show this analytically for linear masked autoencoders and experimentally on masked autoencoders trained using vision transformers. In summary, we expect that there is no silver bullet objective for self-supervised learning for all tasks. Different objectives are more suitable for different tasks, and one must ideally build an ensemble that spans the space of tasks.

3.1 Deep reference priors: Training Model Zoos on unlabeled data

In this section, we turn to the theory of reference priors to train a zoo of models from just unlabeled data. In short, instead of training a single model from unlabeled data, deep reference priors train an ensemble of diverse models that span manifold of probability distributions. In this section:

(1) We introduce the theory of reference priors, which are objective, uninformative Bayesian priors computed by maximizing the mutual information between the task and the weights. We show how these priors maximize the KL-divergence between the posterior computed from the task and the prior, on average over the distribution of the unknown future data. This allows the samples from the task to maximally influence the posterior.

(2) We formalize semi-supervised learning as computing a reference prior where the learner is given access to a pool of unlabeled data and seeks to compute a prior using this data. We show that techniques such as consistency regularization and entropy minimization which are commonly used in practice can be directly understood using the reference prior formulation.

(3) Finally, we show an empirical study of our formulations on the CIFAR-10 and CIFAR-100 datasets. We show that our methods to compute reference priors provide results that are competitive with state of the art methods for semi-supervised learning, e.g., we obtain an **accuracy of 85.45% on CIFAR-10 with 5 labeled samples/class**. We obtain significantly better accuracy than well-tuned fine-tuning for transfer learning, even for very small sample sizes.

3.1.1 Methods

3.1.1.1 Setup

Consider a dataset $\hat{P}_n = \left\{(x_i, y_i)\right\}_{i=1}^n$ with n samples that consists of inputs $x_i \in \mathbb{R}^d$ and labels $y_i \in \{1, \ldots, C\}$. Each sample of this dataset is drawn from a joint distribution $P(x, y)$ which we define to be the "task". We will use the shorthand $x^n = (x_1, \ldots, x_n)$ and $y^n = (y_1, \ldots, y_n)$ to denote all inputs and labels. Let $w \in \mathbb{R}^p$ be the weights of a probabilistic model which evaluates $p_w(y \mid x)$. We will use a random

variable z with a probabilistic model $p_w(z)$ when we do not wish to distinguish between inputs and labels.

Given a prior on weights $\pi(w)$, Bayes law gives the posterior $p(w \mid x^n, y^n) \propto p(y^n \mid x^n, w)\pi(w)$. The Fisher Information Matrix (FIM) $g \in \mathbb{R}^{p \times p}$ has entries $g(w)_{kl} =$

$$\frac{1}{n} \sum_{i=1}^{n} \sum_{y=1}^{C} p_w(y \mid x_i) \partial_{w_k} \log p_w(y \mid x_i) \partial_{w_l} \log p_w(y \mid x_i).$$

It can be used to define the Jeffreys prior $\pi_J(w) \propto \sqrt{\det g(w)}$. Jeffreys prior is reparameterization invariant, i.e., it assigns the same probability to a set of models irrespective of our choice of parameterization of those models. It is an uninformative prior, e.g., it imposes some generic structure on the problem (reparameterization invariance).

3.1.1.2 Reference Priors

Bernardo (1979) suggested that uninformative priors should maximize some divergence, say the Kullback-Leibler (KL) divergence

$$KL(p(w \mid z), \pi(w)) = \int \mathrm{d}w \, p(w \mid z) \log \left(p(w \mid z) / \pi(w) \right),$$

between the prior and the posterior for the data z.

This may seem absurd, since the prior is selected without any knowledge of the distribution of z. The reference prior is computed without knowing the true distribution $p(z)$. Instead, we draw samples z from a distribution hallucinated by the model, i.e., $p(z) = \int \mathrm{d}w \, p(z \mid w)\pi(w)$.

The rationale for this objective is to allow the data to dominate the posterior rather than our choice of the prior. Since we do not know the data *a priori* when selecting the prior, we should maximize the *average* KL-divergence over the data distribution $p(z)$. This amounts to maximizing the mutual information

$$\begin{aligned} \pi^* &= \operatorname*{argmax}_{\pi} I_\pi(w; z) \\ &:= \int \mathrm{d}z \, \mathrm{d}w \, p(z) p(w \mid z) \log \frac{p(w \mid z)}{\pi(w)} = H(w) - H(w \mid z) \end{aligned} \tag{3.1}$$

where $p(z) = \int \mathrm{d}w\, \pi(w) p(z \mid w)$ and $H(w) = -\int \mathrm{d}w\, \pi(w) \log \pi(w)$ is the Shannon entropy; the conditional entropy $H(w \mid z)$ is defined analogously. Mutual information is a natural quantity to measure the amount of missing information on w provided by the data z if the initial belief was π. The prior $\pi^*(w)$ is known as a reference prior. It is invariant to a reparameterization of the weight space because mutual information is invariant to reparameterization. The reference prior does not depend upon the samples $\hat{P}_n$ but only depends on their distribution P.

The objective to calculate reference prior π^* above may not be analytically tractable since z is an infinite-dimensional data vector. Therefore Bernardo also suggested computing n-reference priors. We call n the "order" and deliberately overload the notation for the number of samples n; the reason will be clear soon.

$$\pi_n^* = \mathrm{argmax}_\pi\, I_\pi(w; z^n) = H(w) - H(w \mid z^n), \tag{3.2}$$

using n samples and then setting $\pi^* := \lim_{n\to\infty} \pi_n^*$ under appropriate technical conditions (Berger et al., 1988). Reference priors are asymptotically equivalent to Jeffreys prior for one-dimensional problems. In general, they differ for multidimensional problems, but it can be shown that Jeffreys prior is the continuous prior that maximizes the mutual information (Clarke and Barron, 1994).

3.1.1.3 Blahut-Arimoto algorithm

The Blahut-Arimoto algorithm (Arimoto, 1972; Blahut, 1972) is a method for maximizing functionals and leads to iterations of the form $\pi^{t+1}(w) \propto \exp\left(\mathrm{KL}(p(z \mid w), p(z))\right) \pi^t(w)$. It is typically implemented for discrete variables, for example, in the Information Bottleneck (Tishby et al., 1999). In this case, maximizing mutual information is a convex problem, and therefore the BA algorithm is guaranteed to converge. Such discretization is difficult for high-dimensional deep networks. We therefore implement the BA algorithm using particles; see remark 15.

Example 14 (Estimating the bias of a coin)**.** *To ground intuition, consider the estimation of the bias of a coin* $w \in [0, 1]$ *using* n *trials. If* z *denotes the number of heads (which is a sufficient statistic), we have* $p(z \mid w) = w^z (1-w)^{n-z} n!/(z!(n-z)!)$*. For* $n = 1$*, since we know that* $I(w; z^1) \leq \log 2$ *with this one bit of information, we can see that* $\pi_1^*(z) = (\delta(w) + \delta(1-w))/2$ *is the reference prior that achieves this upper bound. This result is intuitive: If we* know *that we have only one observation, then the optimal uninformative*

prior should put equal probability mass on the two exhaustive outcomes $w = 0$ (heads) and $w = 1$ (tails). We can numerically calculate π_n^* for different values of n using the BA algorithm (fig. 3.1).

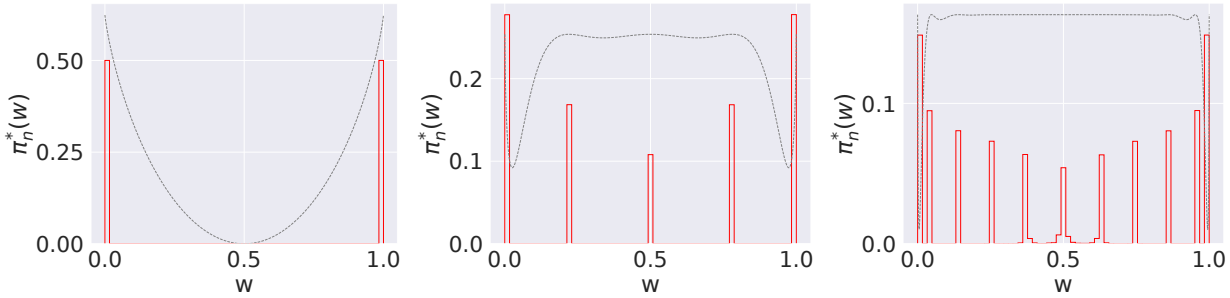


Figure 3.1: We calculated the **reference prior for the coin-tossing model** for $n = 1, 10, 50$ (from left to right) using the Blahut-Arimoto algorithm. Atoms are critical points of the gray line which is $\mathrm{KL}(p(z^n), p(z^n \mid w))$. The prior is discrete for finite order $n < \infty$ (Mattingly et al., 2018). Atoms of the prior are maximally different from each other, e.g., for $n = 1$, they are on opposite corners of the parameter space. As the number of samples increases, the separation between atoms of the prior reduces. The prior converges to Jeffreys prior $\pi_J(w) \propto (w(1-w))^{-1}$ as $n \to \infty$.

We next discusses a key property of reference priors that enables us to calculate them numerically, namely that they are supported on a discrete set in the weight space (section 3.1.1.4). It then formulates reference priors for semi-supervised (section 3.1.1.6) and transfer learning (sections 3.1.1.7 and 3.1.1.8).

3.1.1.4 Existence and discreteness of reference priors

Rigorous theoretical development of reference priors has been done in the statistics literature. We focus on their applications. We however mention some technical conditions under which our development remains meaningful.

A reference prior does not exist if $I_\pi(w; z^n)$ is infinite (Berger et al., 1988). For the concept of a reference prior to remain meaningful, we make the following technical assumptions. (i) π is supported on a compact set $\Omega \subset \mathbb{R}^p$, and (ii) if $p_\pi(z^n) = \int_\Omega \mathrm{d}w\, \pi(w) p(z^n \mid w)$ is the marginal, then $\mathrm{KL}(p_w, p_\pi)$ is a continuous function of w for any π. Under these conditions, the n-order prior π_n^* exists and $I_{\pi_n}(w; z^n)$ is finite; see (Zhang, 1994, Lemma 2.14). Now assume that π_n^* exists and is unique up to a set of measure zero. Let $\Omega_n = \{w \in \Omega : \pi_n^*(w) > 0\}$ be the support of π_n^* and z^n be a discrete random variable with C atoms. If $\{p(z^n \mid w) : w \in \Omega_n\}$ is compact, then π_n^* is discrete with no more than C atoms (Zhang, 1994, Lemma 2.18)).

Remark 15 (Blahut-Arimoto algorithm with particles)**.** *Since the optimal prior is discrete, we can maximize the mutual information directly by identifying the best set of atoms. We set the prior to have the form $\pi_n^* =$*

$\sum_{i=1}^{K} K^{-1}\delta(w - w^i)$ *where* $\{w^1, \ldots, w^K\}$ *are the* K *atoms. We call these atoms "particles". Using standard backpropagation, we can then compute the gradient of the objective in eq.* (3.2) *with respect to each particle (note that the gradient of each particle depends upon all other particles).*

3.1.1.5 Visualizing the reference prior for deep networks

One cannot directly visualize the high-dimensional particles w in π_n^*. But we can think of each particle w as representing a probability distribution $f(w) \in \mathbb{R}^{nC}$ given by

$$\left(\sqrt{p_w(y = 1 \mid x_1)}, \sqrt{p_w(y = 2 \mid x_1)}, \ldots, \sqrt{p_w(y = C \mid x_n)}\right).$$

and use a method for visualizing such distributions developed in Quinn et al. (2019a) that computes a principal component analysis (PCA) of such vectors $\{f(w^1), \ldots, f(w^K)\}$ shown in fig. 3.2.

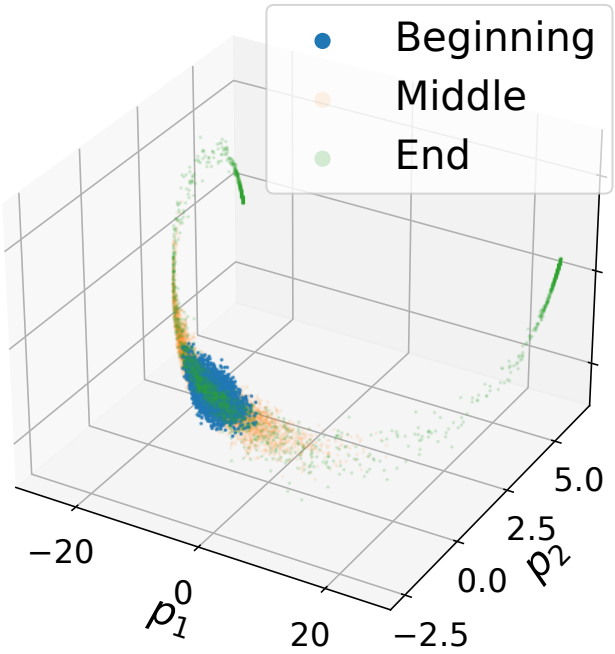


Figure 3.2: Reference prior (green) for binary classification on MNIST. A three-dimensional embedding of the probability distributions of $K = 3000$ atoms in the reference prior after 50,000 iterations of the BA algorithm (green) for a binary classification problem on MNIST (digits 3 vs. 5). Particles were initialized randomly (blue) and they are nearby in this embedding because at initialization, the logits of each particle are uniformly distributed. Orange shows particle locations after 5,000 iterations. As the reference prior objective in eq. (3.2) is optimized, the particles increasingly make more diverse predictions (orange) and towards the end (green) these particles spread apart in the prediction space.

This experiment demonstrates that we can instantiate reference priors for deep networks in a scalable fashion even for a large number of particles K. It provides a visual understanding of how atoms of the prior are

diverse models in prediction space.

How to choose the number of atoms K in the reference prior? Each particle in this paper is a deep network, so must be careful to ensure that we do not maintain an unduly large number of atoms in the prior. Abbott and Machta (2019) suggest a scaling law for K in terms of the number of samples n, e.g., $K \sim n^{4/3}$ for a problem with two biased coins. We will instead treat K as a hyper-parameter. This choice is motivated from the emergent low-dimensional structure of the green particles in fig. 3.2; see the further analysis in in section 3.1.2.1.

Remark 16 (Variational approximations of reference priors)**.** *Nalisnick and Smyth (2017) maximize a lower bound on $I_\pi(w; z)$ and replace the term $p(z) = \int \mathrm{d}w\, \pi(w)p(z \mid w)$ in eq. (3.1) by the so-called VR-max estimator $\max_w \log p(z \mid w)$ where the maximum is evaluated across a set of samples from $\pi(w)$ (Li and Turner, 2016). They use a continuous variational family parameterized by neural networks. However, reference priors are supported on a discrete set. Using a continuous variational family, e.g., a Gaussian distribution, to approximate π_n^* is computationally beneficial but it is detrimental to the primary purpose of the prior, namely to discover diverse models. This is also seen in fig. 3.2 where it would be difficult to construct a variational family whose distributions put mass mostly on the green points. We therefore do not use variational approximations.*

Remark 17 (Reference prior depends upon the number of samples and its atoms are diverse models)**.** *eq. (3.1) encourages the likelihood $p(z^n \mid w)$ of atoms in the reference prior to be maximally different from that of other atoms. This gives us intuition as to why the prior should have finite atoms. Consider the covering number in learning theory (Bousquet et al., 2003) where we endow the model space with a metric that measures disagreement between two hypotheses over n samples. Smaller the number of samples n, smaller the covering number, and smaller the effective set of models considered. The reference prior is similar. If we only have few samples n, then it is not possible for the likelihood in Bayes law to distinguish between a large set of models and assign them different posterior probabilities. The prior therefore puts probability mass only on a finite set of atoms and these atoms have diverse outputs on the n samples. This ability of the prior to select a small set of representative models is extremely useful for training deep networks with few data and it was our primary motivation.*

3.1.1.6 Reference priors for self-supervised learning

Consider the situation where we are **given inputs x^n, their corresponding labels y^n and unlabeled inputs x^u**. Our goal is semi- or self-supervised learning, i.e., to use x^u to build a prior $\pi^*(w)$ that selects models that can be learned using the labeled data (x^n, y^n). Recall that since π^* is a prior, it should not depend on (x^n, y^n). Just like the construction of the reference prior in section 3.1.1.2, we can maximize

$$\begin{aligned} I_\pi(y^n, x^n; w) &= \mathop{\mathbb{E}}_{x^n,(y^n \mid x^n,w),w\sim\pi}\left[\log \frac{p(y^n \mid x^n, w)}{p_\pi(y^n \mid x^n)}\right] \\ &= \mathop{\mathbb{E}}_{x^n,(y^n \mid x^n,w),w\sim\pi}\left[\log p(y^n \mid x^n, w)\right] - \mathop{\mathbb{E}}_{x^n,y^n \mid x^n}\left[\log p_\pi(y^n \mid x^n)\right] \\ &= \mathop{\mathbb{E}}_{x^u}\left[H(y^u \mid x^u)\right] - \mathop{\mathbb{E}}_{x^u,w\sim\pi}\left[H(y^u \mid x^u, w)\right], \end{aligned} \tag{3.3}$$

where $p_\pi(y^n \mid x^n) = \int \mathrm{d}w\, \pi(w) \prod_{i=1}^n p(y_i \mid x_i, w)$ and likewise for $p_\pi(y^u \mid x^u)$. The first step is simply the definition of I_π: it is the KL-divergence of the posterior after seeing (x^n, y^n) with respect to the prior $\pi(w)$. The second step is the key idea, and its rationale is as follows. If we know that the inputs x^u and x^n come from the same task, then we can use samples x^u to compute the expectation over x^n. For the same reason, we can average the outputs y^u which are predicted by the network instead of the fixed labels y^n. We emphasize that both x^u and y^u are averaged out in the objective above. Predictions on new samples x are made using the Bayesian posterior predictive distribution

$$p(y \mid x, x^n, y^n) \propto \int \mathrm{d}w\, \pi_n^*(w) p(y \mid x, w) p(y^n \mid x^n, w). \tag{3.4}$$

An intuitive understanding of eq. (3.3) Assume for now that we know the number of classes C (although the objective is valid even if that is not the case). If our prior has K particles, then the second term is the average of the entropy per particle of the predictions. The objective encourages each particle w_i to predict confidently, i.e., to have a small entropy in its output distribution $p_{w_i}(y \mid x)$. The first term is the entropy of the average predictions: $p_\pi(y^n \mid x^n)$, and it is large if the particles predict different outputs y^n for the same inputs x^n, i.e., they disagree with each other. We treat the constant α (which should be 1 in the definition of mutual information) as a hyperparameter to allow control over this phenomenon. **The reference prior self-supervised learning objective encourages particles to be dissimilar but confident models (not**

necessarily correct).

3.1.1.7 Reference priors for a two-stage experiment

We first develop the idea using generic random variables z^n. Consider a situation when we **see data in two stages, first z^m, and then z^n.** How should we select a prior, and thereby the posterior of the first stage, such that the posterior of the second stage makes maximal use of the new n samples? We can extend the idea in section 3.1.1.6 in a natural way to address this question. We can **maximize the KL-divergence between the posterior of the second stage and the posterior after the first stage, on average, over samples z^n.**

Since we have access to samples z^m, we need not average over them, we can compute the posterior $p(w \mid z^m)$ from these samples given the prior $\pi(w)$. First, notice that $p(w, z^n \mid z^m) = p(w \mid z^{m+n})p(z^n \mid z^m) = p(z^n \mid w)p(w \mid z^m)$. We can now write

$$\begin{aligned}\pi^*_{n \mid m} &= \operatorname*{argmax}_{\pi} I_{p(w \mid z^m)}(w; z^n) \\ &:= \int \mathrm{d}z^n \, p(z^n \mid z^m) \, \mathrm{KL}(p(w \mid z^{m+n}), p(w \mid z^m)) \\ &= \int \mathrm{d}w \, p(w \mid z^m) \int \mathrm{d}z^n \, p(z^n \mid w) \log \frac{p(z^n \mid w)}{p(z^n \mid z^m)},\end{aligned} \tag{3.5}$$

where $p(w \mid z^m) \propto p(z^m \mid w)\pi(w)$ and $p(z^n \mid z^m) = \int \mathrm{d}w \, p(z^n \mid w)p(w \mid z^m)$. The key observation is that if the reference prior eq. (3.2) has a unique solution, we should have that the optimal $p(w \mid z^m) \equiv \pi^*_n(w)$. This leads to

$$\pi^*_{n \mid m}(w) \propto \pi^*_n(w) \, p(z^m \mid w)^{-1}. \tag{3.6}$$

This prior puts *less* probability on regions which have high likelihood on old data z^m whereby the posterior is maximally informed by the new samples z^n. Given knowledge of old data, the prior *downweighs regions* in the weight space that could bias the posterior of the new data. We also have $\pi^*_{n \mid m} = \pi^*_n$ for $m = 0$ which is consistent with eq. (3.2). As $m \to \infty$, this prior ignores the part of the weight space that was ideal for z^m.

Remark 18 (Averaging over z^m in the two-stage experiment)**.** *If we do not know the outcomes z^m yet, the*

prior should be calculated by averaging over both z^m, z^n

$$\begin{aligned}\pi^* &= \underset{\pi}{argmax} \int \mathrm{d}z^m \, p(z^m) I_{p(w \mid z^m)}(w; z^n) \\ &:= I_\pi(w; z^{m+n}) - I_\pi(w; z^m) = H(w \mid z^m) - H(w \mid z^{m+n}).\end{aligned} \tag{3.7}$$

The encourages multiple explanations of initial data z^m*, i.e., high* $H(w \mid z^m)$*, so as to let the future samples* z^n *select the best one among these explanations, i.e., reduce the entropy* $H(w \mid z^{m+n})$*. It is interesting to note that neither is this two-stage prior equivalent to maximizing* $I_\pi(w; z^{m+n})$*, nor is it simply the optimal prior corresponding to objectives* $I_\pi(w; z^m)$ *or* $I_\pi(w; z^n)$*. Both eqs.* (3.6) *and* (3.7) *therefore indicate that two-stage priors are useful when we have* some *data* a priori*, this can be either unlabeled samples from the same task, or labeled samples from some other task.*

Remark 19 (A softer version of the two-stage reference prior)**.** *The objective in eq.* (3.7) *resembles the predictive information bottleneck (IB) of Bialek et al. (2001), or its variational version in Alemi (2020), which seek to learn a representation, say* w*, that maximally forgets past data while remaining predictive of future data*

$$\max_{p(w \mid z^m)} I(w; z^n) - \beta I(w; z^m). \tag{3.8}$$

The parameter β *in eq.* (3.8) *gives this objective control over how much information from the past is retained in* w*. We take inspiration from this and construct a variant of eq.* (3.6)

$$\begin{aligned}\pi^{\beta}_{n \mid m}(w) &\propto \pi^*_n(w) p(z^m \mid w)^{-\beta} \quad \text{for } \beta \in (0, 1). \\ \implies p(w \mid z^{m+n}) &\propto p(z^n \mid w) p(z^m \mid w)^{1-\beta} \pi^*_n(w).\end{aligned} \tag{3.9}$$

We should use $\beta = 0$ *when we expect that data from the first stage* z^m *is similar to data* z^n *from the second stage. This allows the posterior to* benefit *from past samples. If we expect that the data are different, then* $\beta = 1$ *ignores regions in the weight space that predict well for* z^m*. This is similar to the predictive IB where a small* β *encourages remembering the past and* $\beta = 1$ *encourages forgetting.*

3.1.1.8 Reference priors for transfer learning

Consider the two-stage experiment where in the first stage we obtain m samples (x_s^m, y_s^m) from a "source" task P^s and the second stage consists of n samples (x_t^n, y_t^n) from the "target" task P^t. Our goal is to calculate a prior $\pi(w)$ that best utilizes the target task data.

Bayesian inference involves first computing the posterior $p(w \mid x_s^m, y_s^m) \propto p(y_s^m \mid w, x_s^m)\pi(w)$ from the source task and using it as a prior to compute the posterior for the target task $p(w \mid x_t^n, y_t^n, x_s^m, y_s^m)$. Just like section 3.1.1.2, **the key idea again is to maximize the KL-divergence between the two posteriors** $\mathrm{KL}\left(p(w \mid x_t^n, y_t^n, x_s^m, y_s^m),\ p(w \mid x_s^m, y_s^m)\right)$, but averaged over samples x_s^m and x_t^n.

Case 1: Access to unlabeled data from the source x_s^m and the target task x_t^n We should average the KL-divergence over both the source and target predictions y_s^m and y_t^n and maximize

$$\mathop{\mathbb{E}}_{x_s^m, x_t^n, y_s^m \mid x_s^m, y_t^n \mid x_t^n} \mathrm{KL}\left(p(w \mid x_t^n, y_t^n, x_s^m, y_s^m), p(w \mid x_s^m, y_s^m)\right) \tag{3.10}$$

over the prior π. Here $p_\pi(y_s^m \mid x_s^m) = \mathbb{E}_{w\sim\pi}\, p(y_s^m \mid x_s^m, w)$ and $p_\pi(y_t^n \mid x_t^n) = \mathbb{E}_{w\sim\pi}\, p(y_t^n \mid x_t^n, w)$, respectively. Note that averages over x_s^m and x_t^n are computed using samples while averages over $y_s^m \mid x_s^m$ and $y_t^n \mid x_t^n$ are computed using the model's predictions.

Case 2: x_s^m, y_s^m are fixed and known, and we have a pool of unlabeled target data x_t^n Since we already know the labels for the source task, we will only average over x_t^n and y_t^n and maximize

$$\mathop{\mathbb{E}}_{x_t^n, y_t^n \mid x_t^n} \mathrm{KL}\left(p(w \mid x_t^n, y_t^n, x_s^m, y_s^m), p(w \mid x_s^m, y_s^m)\right); \tag{3.11}$$

here $p_\pi(y_t^n \mid x_t^n) = \int \mathrm{d}w\, \pi(w) p(y_t^n \mid x_t^n, w)$.

Remark 20 (Connecting eqs. (3.10) and (3.11) to practice)**.** *Both objectives can be written down as*

$$\pi^* = \mathop{argmax}_{\pi} I_\pi(w; y_t^n, x_t^n, x_s^m, y_s^m) - I_\pi(w; x_s^m, y_s^m) \tag{3.12}$$

with the distinction that while in Case 1, we average over all quantities, namely $p(x_s^m), p(y_s^m), p(x_t^n), p(y_t^n)$ while in Case 2, we fix x_s^m and y_s^m to the provided data from the source task. Case 2 is what is typically called

transfer learning. Case 1, where one has access to only unlabeled data *from a source task* that is different from the target task *is not typically studied in practice. Like eq.* (3.9)*, we can again introduce a coefficient* β *on the second term in eq.* (3.12) *to handle the relatedness between source and target tasks.*

3.1.1.9 Practical tricks for implementing reference priors

The reference prior objective is difficult to implement directly using deep networks and modern datasets. We next discuss some practical tricks that we have developed.

(1) Order of the reference prior n versus the number of samples Bernardo (1979) set the order of the prior n to be the same as the number of samples. We observe that both do not have to be identical and make a distinction between the two. In our expierments, we restrict the order to $n = 2, 3$. Mathematically, this amounts to computing averages in eq. (3.2) or eq. (3.3) over only sets of n-tuples. This significantly reduces the class of models considered in the reference prior by *pretending* that there is a small number of samples available for training the task—which is useful, and also true in practice, for over-parametrized deep networks. This choice is also motivated by the low-dimensional structure in the reference prior in fig. 3.2. Note that we are *not* restricting to small order n for computational reasons, i.e., computing the expectation over all classes y^n in eq. (3.3) can be done in a single forward pass.

(2) Cross-entropy loss to bias particles towards good parts of the weight space The posterior eq. (3.4) suggests that we should first compute the prior, and then weight each particle by the likelihood of the labeled data. In practice, we combine these two steps into a single objective

$$\max_{\pi} \gamma I_{\pi}(w; y^u, x^u) + \mathop{\mathbb{E}}_{w\sim\pi}\left[\log p(y^n \mid x^n, w)\right], \tag{3.13}$$

where γ is a hyper parameter, x^n, y^n are labeled samples. eq. (3.13) allows us to directly obtain particles that both have high probability under the prior and a high likelihood. This is different from the correct Bayesian posterior (which would set $\gamma = 1$, we use $\gamma = 1/2$) but it is a trick often employed in the SSL literature. The second term restricts the search space for the particles in $\pi(w)$.

(3) Data augmentation State of the art SSL methods use a large set of data augmentations, e.g., RandAugment (Cubuk et al., 2020) and CTAugment (Berthelot et al., 2019a), which both have about 20 transformations. Some are weak augmentations such as mirror flips and crops while some others are strong augmen-

tations such as color jitter. Methods such as FixMatch (Sohn et al., 2020) or MixMatch (Berthelot et al., 2019b) use weak augmentations to get soft labels for predictions on strong augmentations.

We compute the entropy term $H(y^u \mid x^u, w)$ in eq. (3.3) using the conditional distribution $p_G(y \mid x, w) = \mathbb{E}_{g\sim G}[p_w(y \mid g(x), w)]$ where $G = G_1 \cup G_2$ is the set of weak (G_1) and strong (G_2) augmentations. Let $g_i \sim G_i$ be an augmentation and denote $p_{g_i} \equiv p_w(y \mid g_i(x), w)$ for $i \in \{1, 2\}$. In every mini-batch we use $p_G(y \mid x, w) \approx \tau p_{g_1} + (1-\tau)p_{g_2}$ where τ is a hyper-parameter. This gives accuracy that is reasonable (about 87% for 500 samples) but a bit lower than that of the state-of-the-art SSL methods. We noticed that if we use an upper bound on the entropy from Jensen's inequality

$$-\mathop{\mathbb{E}}_{x^u} \int \mathrm{d}y^u \, p_G(y^u \mid x^u, w) \left[\tau \log p_{g_1} + (1-\tau) \log p_{g_2}\right] \tag{3.14}$$

then we can close this gap in accuracy (see table 3.1). This is perhaps because the cross-entropy terms, e.g., $-p_{g_1} \log p_{g_2}$, force the predictions of the particles to be consistent across both types of augmentations, just like the objective in FixMatch or MixMatch. Our formulation is thus useful not only to understand SSL but also to tweak it to perform as well as current methods and thereby shed light on the theoretical underpinnings of their performance.

(4) Computing $H(y^u \mid x^u, w)$ A number of SSL methods work by creating pseudolabels from weakly augmented data, which seems to be a key ingredient of good accuracy in our experience with these methods. We tried two heuristics to compute the entropy term $H(y^u \mid x^u, w)$ that are motivated by these papers. First, we follow FixMatch and only use unlabeled data with confident predictions to compute $H(y^u \mid x^u, w)$. A datum x contributes to the objective only if $\max_y p_w(y|g_1(x), w) > 0.95$. Changing this threshold does not lead to deterioration of the accuracy, so this heuristic need not be used while building the reference prior. Second, if G_1 is the set of weak augmentations (see previous point), methods like FixMatch and MixMatch use $\operatorname{argmax}_y p(y \mid g_1(x), w)$ as a pseudo-label but do not update this using the back-propagation gradient. This prevents the more reliable predictions on G_1 from changing. As a result, the entropy term $-\tau^2 p_{g_1} \log p_{g_1}$ is a constant in eq. (3.14). To normalize the terms coming from τ in eq. (3.14), we set γ in eq. (3.13) to $1/(1-\tau^2)$ instead of 1. We have also developed an argument to choose the appropriate value of $\tau = 1/3$. Our experiments suggest that the second heuristic seems essential; we obtain only 10% accuracy

without this heuristic.

3.1.2 Empirical Study

3.1.2.1 Setup

We evaluate on CIFAR-10 and CIFAR-100 (Krizhevsky, 2009). For SSL, we use 50–1000 labeled samples, i.e., 5–100 samples/class and use the rest of the samples in the training set as unlabeled samples. For transfer learning, we construct 20 five-way classification tasks from CIFAR-100 and use 1000 labeled samples from the source and 100 labeled samples from the target task. All experiments use the WRN 28-2 architecture (Zagoruyko and Komodakis, 2016), same as in Berthelot et al. (2019b).

For all our experiments, the reference prior is of order $n = 2$ and has $K = 4$ particles. We run all our methods for 200 epochs, with $\tau = 1/3$ in eq. (3.14) and $\alpha = 0.1$ in eq. (3.3). We set $\gamma = (1 - \tau^2)^{-1}$ as discussed in section 3.1.1.9. For inference, each particle maintains an exponential moving average (EMA) of the weights (this is common in SSL (Tarvainen and Valpola, 2017)).

Semi-supervised learning

Baselines We compare to methods such as FixMatch (Sohn et al., 2020), DASH (Xu et al., 2021), MixMatch (Berthelot et al., 2019b), SelfMatch (Kim et al., 2021), Mean Teacher (Tarvainen and Valpola, 2017), Virtual Adversarial Training (Miyato et al., 2018), and Mixup (Berthelot et al., 2019b).

Table 3.1 compares the accuracy of different SSL methods on CIFAR-10. We find that the reference prior approach is competitive with a number of existing methods, e.g., it is remarkably close to FixMatch on all sample sizes (notice the error bars). There is a gap in accuracy at small sample sizes (40–50) when compared to recent methods. It is important to note that these recent methods employ a number of additional tricks, e.g., FlexMatch implements curriculum learning on top of FixMatch, DASH and FlexMatch use different thresholding for weak augmentations (this increases their accuracy by 2-5%), SelfMatch has higher accuracies because of a self-supervised pretraining stage, FixMatch (CTA) outperforms its RA variant by 1.5% which indicates CTA augmentation is beneficial (we used RA). It is also extremely expensive to train SSL algorithms for 1000 epochs (all methods in table 3.1 do so), we trained for 200 epochs.

Method	Samples				
	50	100	250	500	1000
Mixup	-	-	52.57	63.86	74.28
VAT	-	-	63.97	73.89	81.32
Mean Teacher	-	-	52.68	57.99	82.68
MixMatch	64.21*	80.29*	88.91*	90.35*	92.25*
FixMatch (RA)	86.19 ± 3.37 (40)	90.12*	94.93 ± 0.65	93.91*	94.3*
FixMatch (CTA)	88.61 ± 3.35 (40)	-	94.93 ± 0.33	-	-
DASH (RA)	86.78 ± 3.75 (40)	-	95.44 ± 0.13	-	-
DASH (CTA)	90.84 ± 4.31 (40)	-	95.22 ± 0.12	-	-
SelfMatch	93.19 ± 1.08 (40)	-	95.13 ± 0.26	-	-
FlexMatch	95.03 ± 0.06 (40)	-	95.02 ± 0.09	-	-
Deep Reference Prior	85.45 ± 2.12	88.53 ± 0.67	92.13 ± 0.39	92.94 ± 0.22	93.48 ± 0.24

Table 3.1: Classification accuracy of different semi-supervised learning methods on CIFAR-10. Note: RA and CTA in the methods column indicate that RandAugment or CTAugment were used for augmentations. Entries with * were evaluated by us using open-source implementations from the original authors for 256 epochs. All other entries are from original papers. Entries with "(40)" indicate that 40 labeled samples were used instead of 50.

This experiment shows that our approach to SSL can obtain results that are competitive to sophisticated empirical methods without being explicitly formulated to enforce properties like label consistency with respect to augmentations. This also indicates that reference priors could be a good way to explain the performance of these existing methods, which is one of our goals in this paper.

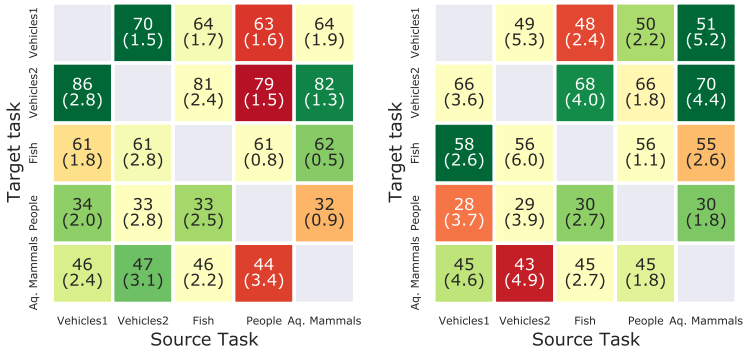
Transfer learning

Just like we did in section 3.1.1.9 for SSL, we instantiate eq. (3.9) and eq. (3.11), by combining prior selection, pretraining on the source task and likelihood of the target task, into one objective,

$$\gamma I_\pi(w; y_t^u, x_t^u) + \mathop{\mathbb{E}}_{w\sim\pi}\left[\log p(w, y_t^n \mid x_t^n)\right] + (1-\beta)\mathop{\mathbb{E}}_{w\sim\pi}\left[\log p(w, y_s^m \mid x_s^m)\right], \tag{3.15}$$

where $\gamma = 1/2$ and $\beta = 1/2$ are hyper-parameters, (x_s^m, y_s^m) are labeled data from the source task ($m = 1000$), (x_t^n, y_t^n) are labeled data from the target task ($n = 100$) and x_t^u are unlabeled samples from the target task (all other samples).

Baselines We use three methods: (a) fine-tuning, which is a very effective strategy for transfer learning (Dhillon et al., 2020; Kolesnikov et al., 2020) but it cannot use unlabeled target data, (b) using only labeled target data (this is standard supervised learning), and (c) using only labeled and unlabeled target data without any source data (this is simply SSL, or $\beta = 1$ in eq. (3.15)). fig. 3.3 compares the performance for pairwise transfer across 5 tasks from CIFAR-100. Our reference prior objective in eq. (3.15) obtains much better accuracy than fine-tuning which indicates that it leverages the unlabeled target data effectively. For each task, the accuracy is much better than both standard supervised learning and semi-supervised learning using our own reference prior approach eq. (3.13); both of these indicate that the labeled source data is being used effectively in eq. (3.15).



Method Task (→)	Vehicles-1	Vehicles-2	Fish	People	Aq. Mammals
Supervised Learning	42.2	63.2	56.8	31.0	42.6
Deep Reference Prior (SSL)	63.6	75.2	54.6	34.0	47.4

Figure 3.3: Top: Accuracy (%) of deep reference priors (left) and fine-tuning (right) for transfer learning tasks in CIFAR-100. Cells are colored red/green relative to the median accuracy of each row. Darker shades of green indicate that the source task is more suitable for transfer. For example, Vehicles-1 as source is the best for all tasks according to the reference prior (left) (which is optimal in theory) but fine-tuning cannot replicate this. The accuracy of cells in the left panel is better than the corresponding cells on the right, e.g., the gap in accuracy is 34.8% for Vehicles 2 → Vehicles 1. **Bottom: Accuracy (%) of supervised learning and SSL for all 5 tasks**. Each number here should be compared to the corresponding row of the matrices in the top panel, e.g., Vehicles 2 has 86% accuracy when transferred from Vehicles 1 using our transfer method (left), it has 66% accuracy from fine-tuning (right), while the same task achieves 63.2% accuracy when trained by itself using supervised learning (table first row) and 75.2% accuracy when trained using unlabeled target data (table second row). Therefore the reference prior-based transfer objective can leverage both labeled source data as well as unlabeled target data. This pattern is consistent for all tasks.

Ablation and analysis

This section presents ablation and analysis experiments for SSL on CIFAR-10 with 1000 labeled samples. We study the reference prior for different settings (i) varying the order n of the prior, (ii) varying the number

of particles in the BA algorithm (K), (iii) exponential moving averaging of the weights for each particle. We also study the two entropy terms in the reference prior objective individually.

We use a reference prior of order $n = 2$ in all our experiments. We see in table 3.2 that **changing the order of the prior** leads to marginal (about 1%) changes in the accuracy.

Method	Order ($\rightarrow$)	2	3	4	5
Deep Reference Prior ($K = 4$)		91.76	90.53	91.51	91.36

Table 3.2: The order of the reference prior has a minimal impact on the accuracy.

Method	#Particles ($\rightarrow$)	2	4	8	16
Deep Reference Prior ($n = 2$)		91.3	91.76	89.79	90.72

Table 3.3: Number of particles has a minimal impact on accuracy.

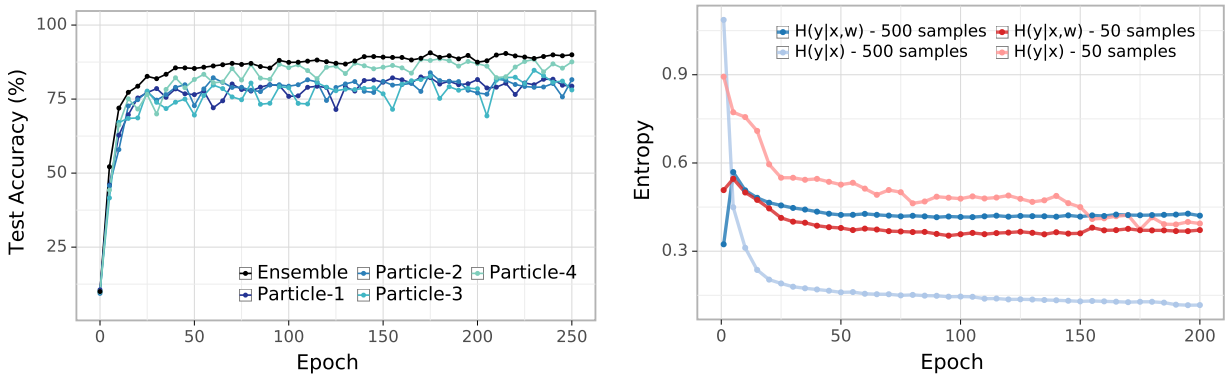


Figure 3.4: **(Left)** Accuracy of individual particles in the prior during training (250 labeled samples). The individual particles have diverse predictions due to the entropy term $H(y^n \mid x^n)$, the accuracy of the ensemble is larger than the accuracy of any single particle. **(Right)** Evolution of entropy terms $H(y^u \mid x^u, w)$ and $H(y^u \mid x^u)$ for two cases (500 labeled samples and 50 labeled samples). While $H(y^u \mid x^u)$ is expected to be larger than $H(y^u \mid x^u, w)$ in eq. (3.3) since KL-divergence is non-negative, this is not always the case since we approximate $H(y^u \mid x^u, w)$ by an upper-bound obtained from Jensen's inequality for data augmentation as discussed in section 3.1.1.9.

We next **vary the number of particles** in the prior in table 3.3 and find that the accuracy is relatively consistent when the number of particles varies from $K = 2$ to $K = 16$. This seems surprising because a reference prior ideally should have an infinite number of atoms, when it approximates Jeffreys prior. We should not a priori expect $K = 2$ particles to be sufficient to span the prediction space of deep networks. But our experiment in fig. 3.2 provides insight into this phenomenon. It shows that the manifold of diverse predictions is low-dimensional. Particles of the reference prior only need to span these few dimension and we can fruitfully

implement our approach using very few particles.

Effect of exponential moving averaging (EMA) We use EMA on the weights of each particle (independently). Table 3.4 analyzes the impact of EMA. As noticed in other works in semi-supervised learning (Berthelot et al., 2019b; Sohn et al., 2020), EMA improves the accuracy by 2-3% regardless of the number of labeled samples used.

Method	#Samples (→)	50	100	250	500	1000
EMA		85.45 ± 2.12	88.53 ± 0.67	92.13 ± 0.39	92.94 ± 0.22	93.48 ± 0.24
No EMA		82.36 ± 2.13	85.64 ± 0.43	89.75 ± 0.36	90.06 ± 1.71	91.57 ± 0.25

Table 3.4: Using EMA for weights of each particle is beneficial and improves accuracy by 2-3%.

The two entropy terms in the reference prior objective fig. 3.4 (left) shows how, because of the entropy term $H(y^u \mid x^u)$, the accuracy of particles is quite different during training. Particles have different predictive abilities (7% range in test error) but the Bayesian posterior predictive distribution has a higher accuracy than any of them. fig. 3.4 (right) tracks the two entropy terms in the objective. For large number of labeled data (500, blue) the entropy $H(y^u \mid x^u)$ which should always be higher than $H(y^u \mid x^u, w)$ in eq. (3.3) is lower (this is not the case for 50 samples, red). This is likely a result of the cross-entropy term in the modified objective in eq. (3.13) which narrows the search space of the particles. This experiment is an important insight into the working of existing semi-supervised learning methods as well, all of which also have a similar cross-entropy objective in their formulation. It points to the fact that at large sample-sizes, the cross-entropy loss and not the semi-supervised learning objective could dominate the training procedure.

3.2 Masked reconstruction learns features at different scales

We end this chapter by studying masked reconstruction, the key objective behind most image (He et al., 2021; Xie et al., 2021) and video (Feichtenhofer et al., 2022; Tong et al., 2022; Wang et al., 2023) foundation models. Despite their success, we do not understand how certain design choices for masked autoencoders (MAEs) — such as masking ratio, patch size, and model depth — affect the downstream tasks that we can solve. Our results point to how masked reconstruction is not a silver bullet to self-supervised learning, and there is no single hyperparameter that works for all tasks.

Our central hypothesis is that *MAEs learn representations that capture spatial correlations in the input image* (fig. 3.5), with the masking ratio and the patch size controlling the scale of the learned features. In other words, the size and number of masks control the features learned by an MAE. We propose that MAEs help lear ViT architectures which inherently lack it, . To formalize this hypothesis, we make the following key contributions:

1. We formalize MAEs in the linear setting and derive an analytical expression for the encoder and decoder to characterize the features learnt by an MAE. We show how MAEs can learn features that capture short- or long- range correlations by controlling the masking ratio and patch size.

2. We provide some insights for practitioners on how to select hyperparameters for MAEs such as masking ratio, patch size, and number of encoder/decoder layers.

3.2.1 Linear Masked Autoencoders

Reconstruction is a popular objective for self-supervised learning but it often fails to learn useful features for downstream perception tasks (Balestriero and LeCun, 2024). Masked reconstruction (He et al., 2021; Xie et al., 2021), has emerged as a popular choice for pretraining deep networks. In this subsection, we study how reconstruction differs from masked reconstruction, and answer when the latter is useful for downstream perception tasks.

We first consider a linear MAE and analyze the kind of solutions obtained for different masking ratios and

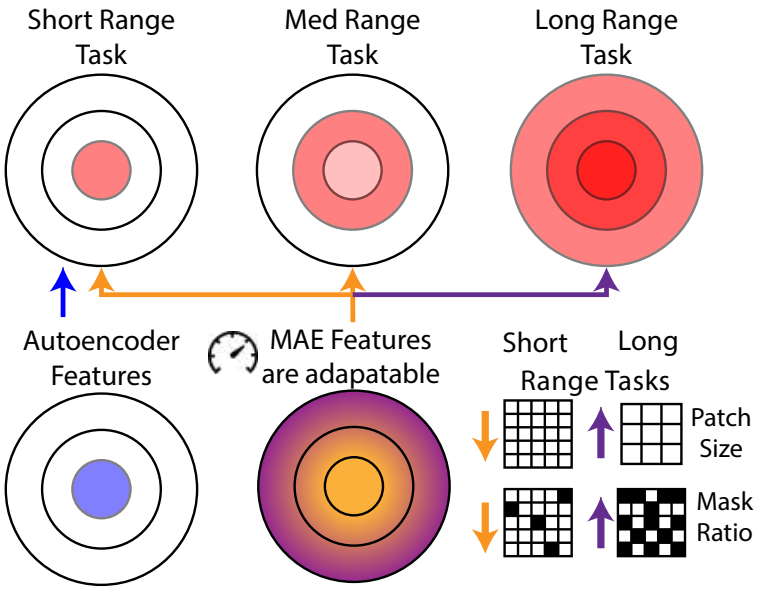


Figure 3.5: **(Top)**: Different tasks have different spatial dependence on the information needed to successfully them, from short-range tasks that only require neighboring information, to long-range tasks that require aggregation of spatially distant information. **(Bottom)**: Autoencoders learn features for one set of tasks, while MAEs can modulate the set of features they learn by changing the masking ratio or patch size, in order to successfully solve different tasks.

Figure 3.6: A $d = 8$ dimensional input can be divided up into 4 patches, each of size $p = 2$.

patch-sizes. Consider input data $X \in \mathbb{R}^{n \times d}$ where n is the number of samples, each being a real-valued vector in d dimensions. Consider a linear encoder with weights $A \in \mathbb{R}^{d \times k}$ and a linear decoder with weights $B \in \mathbb{R}^{k \times d}$—the network has no non-linearities. The encoder projects the input from d input dimensions to k latent dimensions, and the decoder projects the representation back into d dimensions. A linear MAE minimizes the objective

$$\ell_m(A, B) = \mathbb{E}_R \left[\|X - (R \odot X)AB\|^2 \right], \tag{3.16}$$

where $R \in \{0, 1\}^{n \times d}$ is a random variable denoting a mask and $\odot$ is the element-wise product. We can arrange our d-dimensional input vector into patches, each of size p (see fig. 3.6). This is similar to how images are divided into patches in a Vision Transformers (ViT) (Dosovitskiy et al., 2020). Patches are either fully masked or fully visible; this means $R_{ki} = R_{kj}$ if i and j belong to the same patch for any image with index k. Each patch is masked according to a Bernoulli random variable with probability m.[9]

[9] Typical implementations of MAEs mask exactly a fraction m of the patches instead of using a random variable for each patch to

In lemma 1, we compute the expectation in eq. (3.16) in closed form to reduce the linear MAE objective to

$$\ell_m(A, B) = \underbrace{\|X - (1-m)XAB\|^2}_{\text{reconstruction}} + m(1-m)\underbrace{\|GAB\|^2}_{\text{regularizer}}, \tag{3.17}$$

where $G \in \mathbb{R}^{d\times d}$ satisfies $G^\top G = \text{blkdiag}_p(X^\top X)$. The block-diagonal matrix $\text{blkdiag}_p(X^\top X)$ consists of block diagonal entries of size $p \times p$, with the off-diagonal blocks set to zero. This suggests that the objective used to train an MAE consists of two terms. The first term is the standard reconstruction loss of an autoencoder that encourages $X \approx (1-m)XAB$. The second term $\|GAB\|^2$ acts as a regularizer. The masking ratio m controls the strength of the regularizer while the patch-size control the block-diagonal structure of the regularizer. Note that $m = 0$ recovers the objective for a non-masked autoencoder (AE). The regularizer term here therefore is the "bias" of a MAE. It forces the representation of an MAE to deviate from that of a AE.

3.2.1.1 Characterizing the minima of MAEs

Next, we characterize the encoder obtained using a linear MAE (eq. (3.17)). Baldi and Hornik (1989) show that AEs are closely related to principal component analysis (PCA). The encoder at any global minimum extracts the top-k principal components of the data. Linear AEs therefore discover features that best explain the variance in the input data. The training objective of a linear AE does not have local minima. All global minima are of the form $A = UC^{-1}, \quad B = CU^\top$, where the columns of $U \in \mathbb{R}^{d\times k}$ are the top-k eigenvectors of $X^\top X$, and $C \in \mathbb{R}^{k\times k}$ is any invertible matrix.

We can perform a similar analysis for linear MAEs, to derive analytical expressions for the optimal encoder and decoder, and characterize the set of all critical points of this objective. Note that while there are no non-linearities in a linear MAEs, the training objective is still non-convex.

Theorem 21. *Let* $V = (1-m)X^\top X + m\,\text{blkdiag}_p(X^\top X)$ *and let the columns of* U_k *denote the top-k eigenvectors of* $X^\top XV^{-1}X^\top X$. *The objective in eq.* (3.16) *is minimized when the decoder*

$$B = CU_k \tag{3.18}$$

determine if it should be masked. Typically, MAEs also reconstruct only the unmasked patches at the output of the decoder, instead of employing an objective that reconstructs all patches like we have done here.

and the encoder

$$A = V^{-1}X^\top XB^\top(BB^\top)^{-1}C^{-1}, \tag{3.19}$$

where C is any invertible matrix of size $k \times k$.

Every critical point of the MAE objective is a subset of k eigenvectors of $X^\top XV^{-1}X^\top X$ (from lemma 2), i.e., there are exponentially many critical points. The product of the encoder and the decoder for an MAE, comprises of two essential parts. The first part $V^{-1}X^\top X$ whitens the data using a weighted mixture of $X^\top X$ and $\text{blkdiag}(X^\top X)$. The second part $B^\top(BB^\top)^{-1}B$, projects the data onto the column space of B.

Remark 22. *If the input correlation matrix $X^\top X$ is itself block diagonal, i.e., spatial correlations in the input data are non-zero within a patch of size p, then $V = (1-m)X^\top X + m\,\text{blkdiag}_p(X^\top X) = X^\top X$ and which recovers the regular autoencoder solution from theorem 21, with a patch-size of p. In this case, a linear autoencoder and a masked autoencoder have identical minima.*

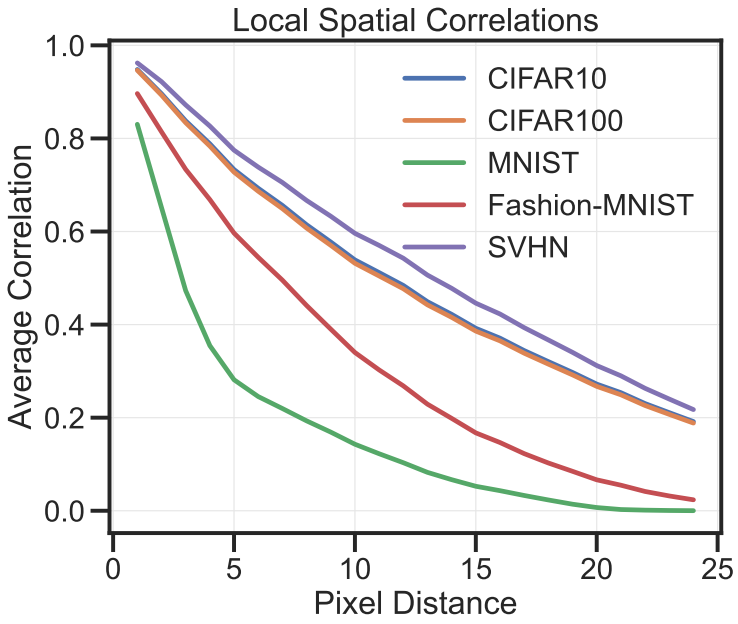


Figure 3.7: Spatial correlations between pixels as a function of the distance between them for different datasets. Correlation between pixels is inversely correlated to the distance for all 5 datasets, i.e., images have stronger local correlations.

Understanding the differences between an MAE and an AE using an Ising model. Images exhibit strong local correlations (fig. 3.7) and the strength of the correlations between pixels is inversely proportional to the distance between them (Hyvärinen et al., 2009). To emulate a part of this structure, we consider data

drawn from an Ising model. This model exhibits strong local correlations controlled by the coupling constant J. Under the Ising model, input $x \in \{-1, 1\}^d$ has probability

$$p(x) \propto \exp\left(Jx_1x_d + J\sum_{i=1}^{d-1} x_ix_{i+1}\right).$$

We fit both a linear AE and a linear MAE on samples from this distribution using objective in eq. (3.16) for the MAE. For this model, we approximate the correlation between x_i and x_j by its estimate in the thermodynamic limit, i.e. $\langle x_i, x_j\rangle \approx \tanh(J)^r$ where $r = \min(|i-j|, d-|i-j|)$ is the distance between x_i and x_j. This can be used to compute $X^\top X$, which can be substituted into the analytical expressions in theorem 21.

In fig. 3.8, we plot the encoder weights (A) and the product of the encoder and decoder weights (AB) for an AE and MAE trained on the Ising model with $d = 32$ dimensions and coupling constant $J = 2$. Both the MAE and AE consider an encoder that projects data from 32 to 6 dimensions. For the MAE, we consider patch-size of 8 and masking ratio of $m = 0.5$. For the AE (fig. 3.8 left), the matrix AB has rank 6 (which is the dimensionality of the feature space). The product AB should be close to identity to minimize the autoencoder objective, and this is indeed evident in the experiment. In fig. 3.8, the encoder of an autoencoder learns features that resemble sinusoids of different frequencies, similar to that of natural images (Hancock et al., 1992).

What is remarkable in this simple experiment is that an MAE learns a different encoder, one that clearly prioritizes input dimensions at the boundary of the patches. Since data has strong local correlations, the boundary of a patch has the largest correlation to nearby patches. The boundary of a patch is therefore most useful for reconstructing other patches. **In summary, MAEs selects features that are redundantly present across patches while autoencoders pick features that best explain the variance in the data.** MAEs are perhaps successful on downstream tasks because many perception tasks are redundant functions of the input (Ramesh et al., 2024), and MAEs find such redundant features.

3.2.1.2 Features of a linear MAE for natural images

We next train on natural images from CIFAR-10 and inspect how linear MAEs, linear denoising autoencoders (DAEs), and AEs encode different kinds of features. Unlike MAEs, which perform masking, DAEs add

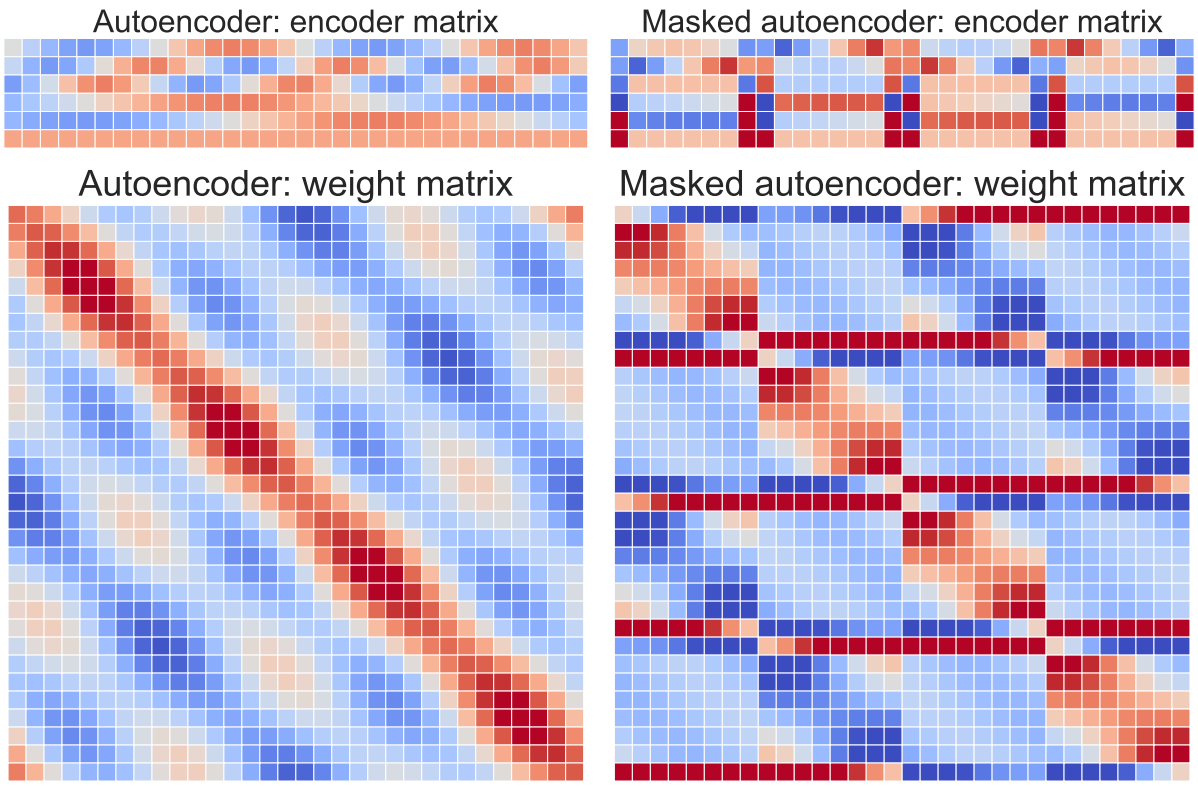


Figure 3.8: We plot the product of the encoder and decoder matrices (bottom) and the encoder weight matrix (top) for an autoencoder (left) and masked autoencoder (right). Red indicates weights with higher magnitude while blue indicates weights with lower magnitude. The plots show that masked autoencoders learn an encoder that projects the data onto a basis that is different from a regular autoencoder. The masked autoencoders prioritizes features at the boundary of the patch since it is most correlated to the other patches and hence most useful for predicting them.

Gaussian random noise to the input (Vincent et al., 2010). For linear encoders and decoder, DAEs eq. (3.20), denoise additive Gaussian noise with variance σ^2, and are equivalent to autoencoders with an ℓ_2 regularization on the weights (AB).

$$\ell_d(A, B) = \mathbb{E}_{L\sim\mathcal{N}(0,\sigma^2)}\left[\|X - (X + L)AB\|_2^2\right] \tag{3.20}$$

For AEs, DAEs and MAEs, we compare the average influence that an input pixel has on an output pixel as a function of the distance between the two. To compute this influence, we consider the Jacobian of the output with respect to the input. For example for our linear models, the quantity $|(AB)_{ij}|$ determines the influence of input pixel i, towards reconstructing target pixel j. We compute the average magnitude of the normalized absolute weight of the entries of the Jacobian as a function of the distance of the output pixel from the input pixel for linear AEs, DAEs and MAEs trained on CIFAR-10 in fig. 3.9. We find that AEs learn more localized

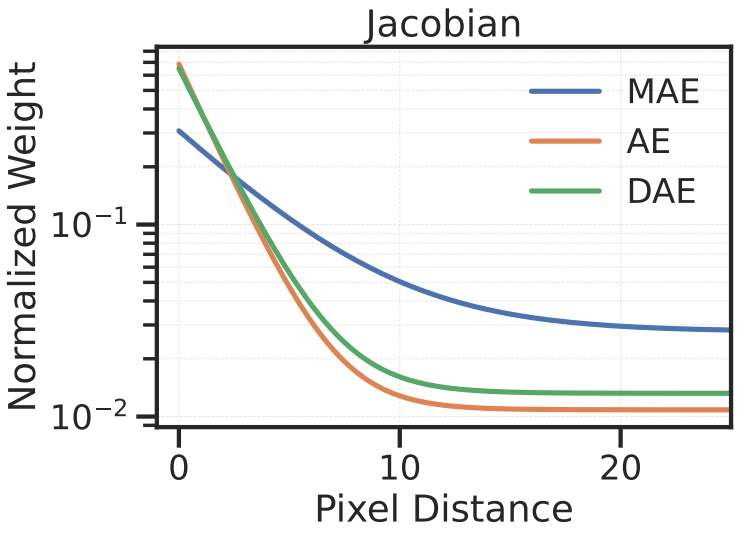


Figure 3.9: An exponential fit to the magnitude of the entries of the input-output Jacobian as a function of distance from the output pixel from the input pixel of different models (averaged over input and output pixels). Results are shown for three methods: Masked Autoencoders (MAEs) with 80% masking ratio and patch size $p = 2$, standard Autoencoders (AEs), and Denoising Autoencoders (DAEs) with noise level $\sigma = 0.2$. Experiments are conducted on CIFAR-10 with a latent dimension of 128. In general, MAEs integrate spatial information from distant patches, as opposed to an AE or a DAE.

kernels and DAEs are slightly less localized than AEs. On the other hand, MAEs integrate spatially distant information. This is because the MAE objective encourages the use of other patches to reconstruct a pixel.

Next, we investigate how MAE hyperparameters—masking ratio m and patch size p—affect the types of learned correlations. Mathematically, we can see this influence from eq. (3.19), where the term $V^{-1}X^\top X$ first performs a whitening transformation of the original data. As the masking ratio increases, this transformation effectively performs patch-wise whitening, making all intra-patch correlation identity and normalizing inter-patch correlations. Masking forces the model to incorporate non-local information beyond the immediate patch for reconstruction, as seen in fig. 3.10. Our results reveal that a high masking ratio produces a more diffuse average Jacobian, whereas a low masking ratio yields a more localized one. We observe a similar relationship with patch size, aligning with our prediction in eq. (3.19): larger patch sizes expand the integration of information from regions outside the patch. In both of these cases, increasing the masking ratio and patch size reduces the weight of intra-patch pixels.

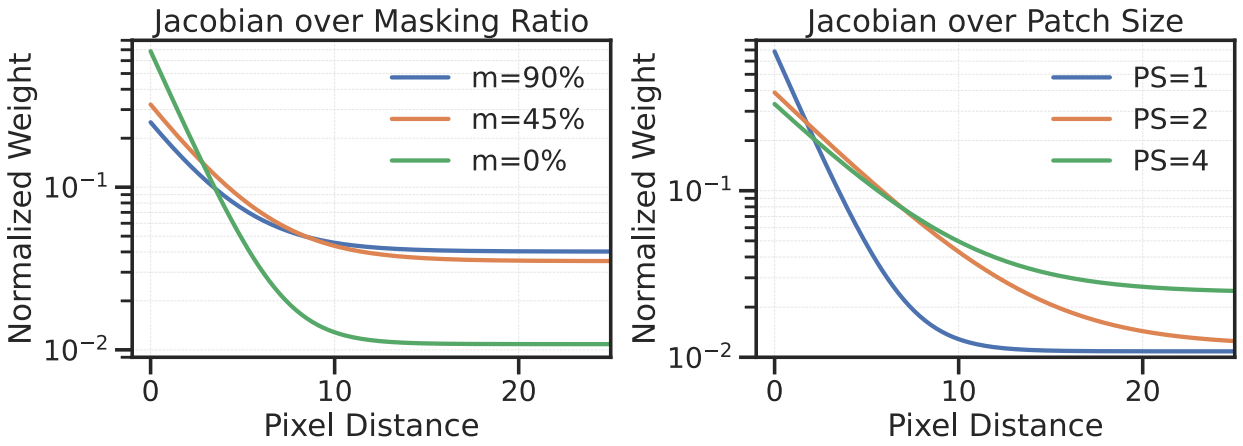


Figure 3.10: An exponential fit to the magnitude of the entries of the input-output Jacobian as a function of distance from the output pixel from the input pixel of different models (averaged over input and output pixels) for different masking ratios and patch sizes. Experiments were conducted for a linear MAE with latent dimension 128 on CIFAR-10. As the masking ratio increases, more and more non-local information is used for reconstruction in a (linear) MAE. Similarly, the reconstruction relies more on nonlocal information as the patch size increases.

3.2.1.3 What kinds of tasks benefit from MAEs compared to AEs?

Previously, we showed that MAEs learn different features than AEs and DAEs, as illustrated in fig. 3.9, e.g., MAEs capture more spatially distant information while reducing the emphasis on intra-patch details. This raises a key question: what kinds of tasks benefit more from MAEs compared to AEs? To investigate this, we first examine a task that explicitly requires integrating spatially distant information, followed by an evaluation of a real-world application—monocular depth prediction using the Cityscapes dataset.

Figure 3.10 suggests that MAEs capture more spatially distant information as the masking ratio and patch size increase. To investigate this, we train MAEs and evaluate them on the task of predicting Gabor features, a task that allows precise control over the spatial dependencies required for accurate prediction. Gabor functions, denoted as $g(i, j)$ for spatial coordinates i, j, are localized harmonics modulated by a Gaussian window and are parameterized by frequency f, phase shift ϕ, Gaussian scale σ, and dilation γ:

$$g(i, j) = \exp\left(-\frac{i^2 + \gamma^2 j^2}{2\sigma^2}\right) \cos\left(2\pi i f + \phi\right). \tag{3.21}$$

Our target, $x_g(i, j)$, is the convolution of the Gabor function over the entire image $x_g(i, j) = (x * g)(i, j)$. The key advantage of using the Gabor feature prediction task is that we can adjust the parameter σ to control

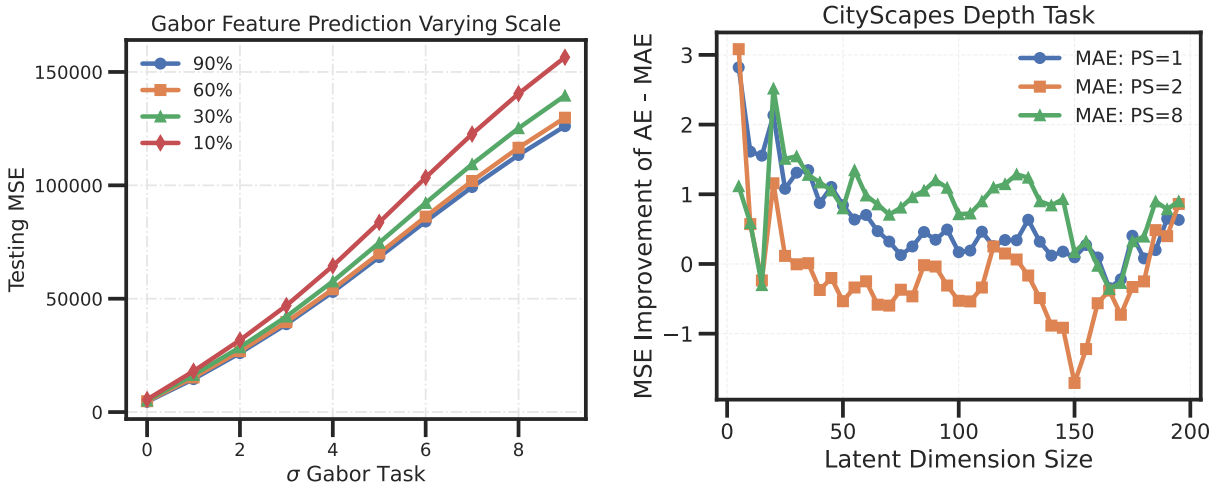


Figure 3.11: Left: Gabor feature prediction task, where we use the pretrained features from MAEs, and showcase their performance as a function of spatial scale (σ), we select. In general, we see a higher masking ratio, which incorporates more non-local information, performs better on more non-local tasks. **Right:** MSE performance reduction of a MAE over an AE as a function of the latent dimension. We use the pretrained MAE or AE features and show how these perform on the task of supervised depth prediction on CityScapes depth dataset.

whether the task depends on local or long-range spatial interactions. This design enables a systematic evaluation of whether MAEs trained with higher masking ratios are better at capturing long-range dependencies. To analyze this effect, we trained an MAE with patch size ($p = 2$) and frequency values $f = [0.03, 0.06, 0.1]$, with fixed $\phi = 0$ and $\gamma = 1$, across various masking ratios (fig. 3.11). For small σ values, different methods perform similarly. However, as σ increases, MAEs trained with high masking ratios outperform those with lower masking ratios. This aligns with our findings in fig. 3.9, where MAEs integrate information from distant spatial patches.

While the Gabor prediction task provides a controlled setting to analyze MAEs, real-world perception tasks present greater complexity. To assess how MAEs perform in practical scenarios, we evaluate them on monocular depth prediction—a task that requires integrating spatial information to infer depth accurately. We trained MAEs on the Cityscapes dataset, downsampled to 32×32 resolution, and analyzed performance as a function of latent space dimensionality. For small latent dimensionalities, all patch sizes ($m = 0.8$) outperform an AE. This finding highlights how MAEs, through different masks, can tailor their representations to different downstream perception tasks.

Linear MAEs are amenable to analysis but they come with their own limitations. (i) On datasets like CIFAR, ViTs use an over-complete representations of the data, while our analysis is restricted to rank-deficient or full-rank encoders and decoders. (ii) Linear MAEs only capture 2nd-order correlations of the data. Higher order correlations in the data distribution can be used for better reconstruction, and possibly provide a better prior for downstream tasks. (iii) The network architecture for linear MAEs is limited to fully connected networks, but it does not give us insight into designing ViTs for MAEs.

3.2.2 From theory to practice: How to train your MAE

Masked autoencoders are notoriously slow to train. Both pretraining and fine-tuning can be sensitive to the choice of hyperparameters, particularly when we use ViTs (Steiner et al., 2021). In this section, we discuss some empirical insights on MAE pretraining and fine-tuning based on our experiments on CIFAR-10.

3.2.2.1 Experimental details

We train MAEs using the architecture in He et al. (2021). We fix the embedding dimension d at 192, the number of attention heads in the encoder to 12 and the decoder to 16 in all our experiments. The Transformer blocks use the GeLU non-linearity (Hendrycks and Gimpel, 2016) with the layer-norm before the attention and MLP blocks — also known as the pre layer-normalization transformer (Xiong et al., 2020).

The MAE is trained for 2000 epochs using the AdamW optimizer with 50 epochs of warmup and a cosine-annealed learning rate schedule with an initial learning rate of 1.2×10^{-3}. The MAEs are trained with a batch-size of 4096 and weight-decay of 0.05. After pretraining, we discard the decoder and fine-tune the encoder. We fine-tune for only 100 epochs using a batch-size of 1024 with 5 epochs of warmup and an initial learning rate of 10^{-3} annealed to 10^{-5} using a cosine-annealed schedule. For MAE pretraining, we use random-resize crop and horizontal flips as the two augmentations, while for fine-tuning we use RandAugment (Cubuk et al., 2020).

We evaluate the accuracy on the downstream task using a linear-probe applied to the encoder. We find that trends obtained using a linear probe are nearly identical to trends obtained using fine-tuning (see section A.5).

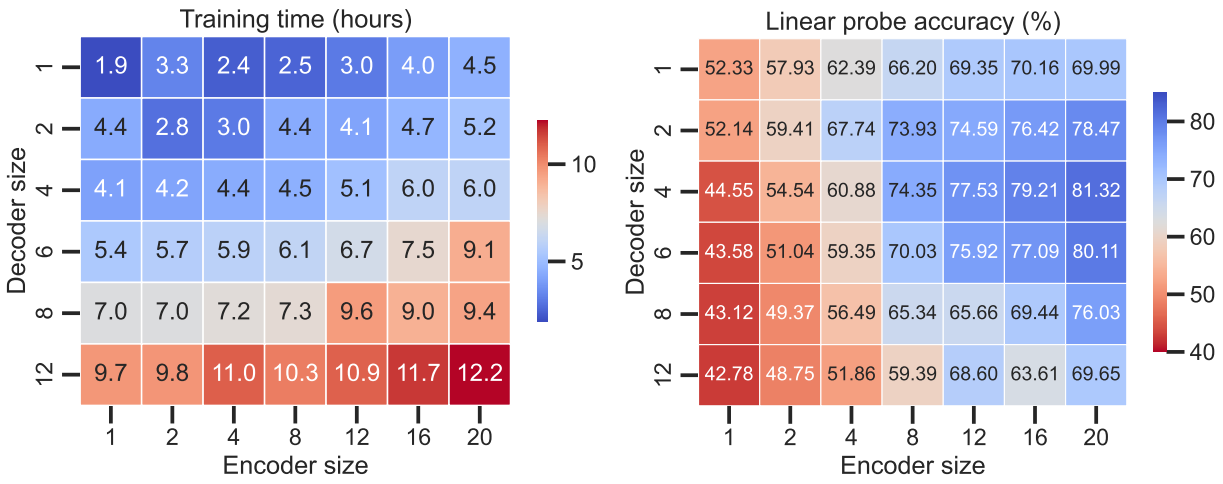


Figure 3.12: We plot (left) the training time and (right) linear probe accuracy for different combinations of the number of encoder and decoder layers. **MAEs are slow to train** with training time increasing with the size of the decoder. We find that the training loss is not a good proxy for downstream classification accuracy (fig. A.20). The **accuracy of the trained encoder continues to improve as we increase its size**. However, this is not true for the decoder: the optimal decoder size is around 2-4 layers. We notice that the differences in accuracies reduce when we fine-tune the encoder for a 100 epochs (fig. A.21).

3.2.2.2 Results

How do we select the size of the encoder and decoder? Masked autoencoders typically have more encoder layers than decoder layers but is this optimal? We investigate how the number of encoder and decoder layers affect the reconstruction error, training time and downstream accuracy for classification. In fig. 3.12, we train models on CIFAR-10 and plot the training time (in hours) when we use a different number of encoder and decoder layers. The training time increases at a slower rate when we increase the number of encoder layers compared to increasing the number of decoder layers since the decoder operates on a longer sequence of tokens — the decoder uses both masked and unmasked tokens while the encoder only uses unmasked tokens. Even for CIFAR-10, training can take as long as 6 to 10 hours on a single GPU.

Increasing the number of encoder and decoder layers decreases the reconstruction error (fig. A.20). However, a smaller reconstruction error does not result in higher linear-probe accuracies (see fig. A.20) or fine-tuning accuracies (see fig. A.21). **We find that the linear-probe accuracy continues to improve as we increase the size of the encoder. However, the optimal decoder has far fewer layers and even a 1-layer decoder has high accuracy after fine-tuning with the added benefit of reduced training time.** For example, a model pretrained with a 1-layer decoder achieves 95.26% accuracy after fine-tuning, which is only 0.30%

worse than our best model fig. A.21.

How do we select the masking ratio and patch-size? The masking ratio and patch-size control the basis learnt by a linear MAE (section 3.2.1). We see that these two hyper-parameters also have a significant impact on MAEs trained using deep networks. In fig. 3.13, we plot the reconstruction error and the linear-probe accuracy for different values of masking ratio and patch-size. **MAEs train faster if the patch-size is large or if the masking ratio is small** since a larger patch-size reduces the number of tokens by a quadratic factor and increasing the masking ratio decreases the number of tokens fed to the encoder. The training time increases from 4 hours to 10 hours as we decrease the masking ratio from 0.9 to 0.1.

Reconstruction improves as we decrease the masking ratio which is unsurprising — it is easier to reconstruct an image if we have more patches. Reconstruction error also reduces as the patch-size becomes smaller. A smaller patch-size reduces the average spatial distance to unmasked patches, which makes reconstruction easier due to strong local spatial correlations seen in images (fig. 3.7). We find that a small patch-size and large masking ratio achieve the best linear probe accuracies. While smaller values of patch-size achieve better linear probe accuracies, an MAE with larger patch-size trains faster.

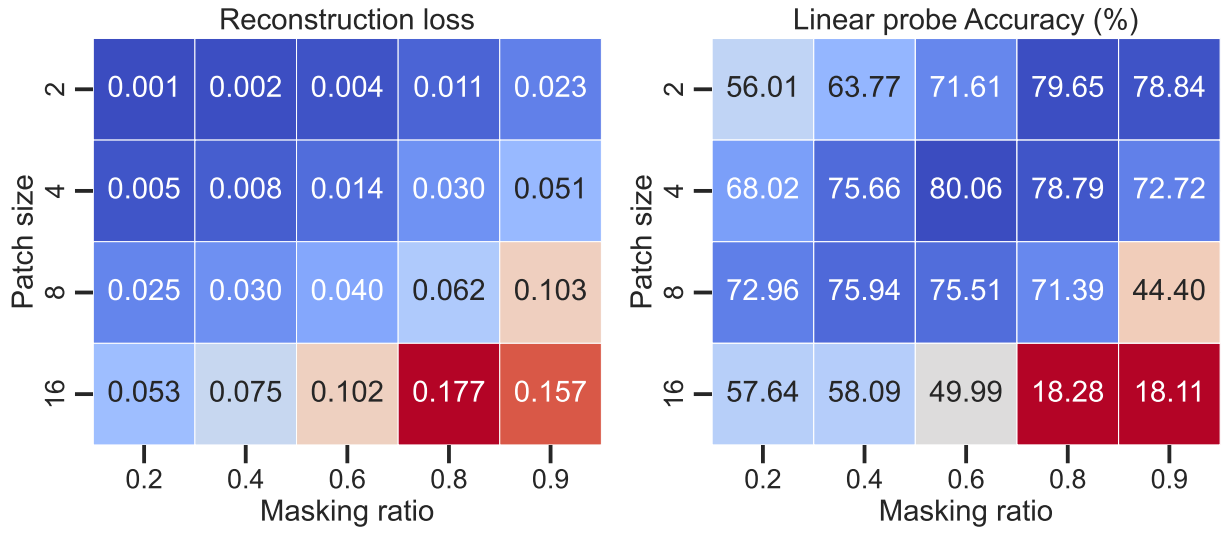


Figure 3.13: We plot the reconstruction error (left) and the linear probe accuracy (right) for different values of patch-size and masking ratio. Reconstruction loss decreases as we decrease the patch-size or decrease the masking ratio. However the linear probe accuracy has no clear trend with respect to the masking ratio and the optimal masking ratio depends on the patch-size. While smaller values of patch-size achieve better linear probe accuracies, they are also slower to train.

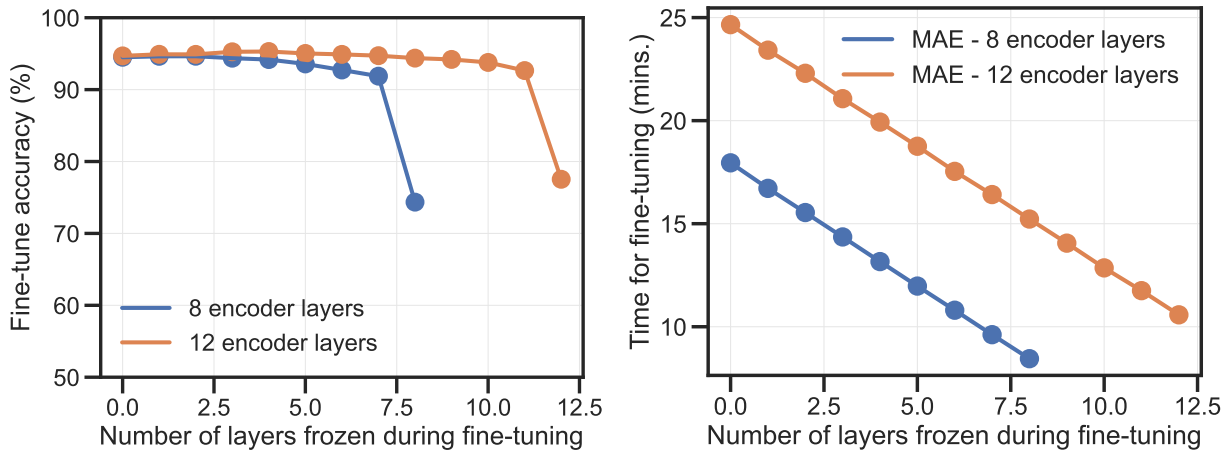


Figure 3.14: We fine-tune the model for 100 epochs after freezing the first k layers of the network and plot the accuracy against number of layers frozen during fine-tuning. We find that even if we freeze all but 1 Transformer block, we recover the accuracy compared to when no layers are frozen, i.e., the accuracy can be recovered while using a fraction of the training time and memory.

Can we fine-tune fewer layers to reduce GPU memory and training time? We investigate the effect of fine-tuning a subset of the layers of an MAE in fig. 3.14. We find that **if we freeze all but one Transformer block, we achieve within 2% of the accuracy achieved from fine-tuning all the layers**, i.e., a similar accuracy can be achieved with a nearly 2x speedup (fig. A.21 (right)) and with less GPU memory. These results align with the practice of setting learning rates that exponentially decrease across the different layers of the encoder during fine-tuning (He et al., 2021).

3.3 Related work

Reference priors in Bayesian statistics We build upon the theory of reference priors which was developed in the objective Bayesian statistics literature (Bernardo, 1979; Berger et al., 1988, 2009). The main idea used in our work is that non-asymptotic reference priors allow us to exploit the finite samples from the task in a fundamentally different way than classical Bayesian inference. If the number of samples from the task available to the learner is finite, then the prior should also select only a finite number of models. Reference priors are not common in the machine learning literature. A notable exception is Nalisnick and Smyth (2017) who optimize a variational lower bound and demonstrate results on small-scale problems. The main technical distinction of our deep reference priors is that we explicitly use the discrete prior instead of a variational approximation.

Self-supervised learning. Contrastive learning (Becker and Hinton, 1992) learns representations without using ground-truth labels (Gutmann and Hyvärinen, 2010; Chen et al., 2020a). It has been extremely effective for self-supervised learning (Doersch and Zisserman, 2017; Kolesnikov et al., 2019), Given inputs (say images) x from P, contrastive learning forces the representation $\varphi(g(x); w_1)$ and $\varphi(g'(x); w_1)$ (shortened to $\varphi(g(x))$ below) of the same input for two different augmentations g, g' to be similar. And forces it to be different from representations of other augmented inputs x' (Zbontar et al., 2021; Bachman et al., 2019; Dosovitskiy et al., 2014).

Semi-supervised learning Our formulation sheds light on the working of current SSL methods. For example, the reference prior can automatically enforce consistency regularization of predictions across augmentations (Tarvainen and Valpola, 2017; Berthelot et al., 2019b), as we discuss in section 3.1.1.9. Similarly, minimizing the entropy of predictions on unlabeled data, either explicitly (Grandvalet et al., 2005; Miyato et al., 2018) or using pseudo-labeling methods (Lee et al., 2013; Sajjadi et al., 2016), is another popular technique. This is automatically achieved by the objective in eq. (3.3). Disagreement-based methods (Zhou and Li, 2010) employ multiple models and use confident models to soft-annotate unlabeled samples for others. Disagreements in our formulation are encouraged by the entropy $H(y^n \mid x^n)$ in eq. (3.3). If $p(y^n \mid x^n)$ is uniform, which is encouraged by the reference prior objective, particles disagree strongly with each other.

Masked autoencoders. Theoretical work on understanding MAEs can be categorized into the following themes: architectural modifications, data modeling, connections to other self-supervised learning objectives, and denoising. Work on architectural modification generally demonstrates that masking alters the attention structure of ViTs, creating more local attention in early layers rather than global (Cao et al., 2022; Xie et al., 2023b; Park et al., 2023; Huang et al., 2024). This is significant as Raghu et al. (2021) suggests that networks with local receptive fields generalize better to vision tasks. Kong et al. (2023) hypothesizes that this learning process attempts to learn a hierarchical latent variable model, where masking enforces identifiability of coarser latent variables, though such a model is not definitively known to exist for natural image data. Another perspective interprets MAEs as a form of contrastive learning using only positive samples of unmasked patches (Zhang et al., 2022). Littwin et al. (2024) challenges this hypothesis by demonstrating that MAE approaches learn a different set of features than contrastive methods. Beyond explaining the architectural bias induced by MAEs, researchers have also investigated how MAEs can perform masked image modeling tasks under extreme masking ratios. This challenge can be framed as a linear inverse problem, where measurements of unmasked pixels are used to recover masked pixels (Kadkhodaie and Simoncelli, 2021). While compressive sensing offers one method to solve this inverse problem, it lacks the nonlinear encoder and prior over the data distribution present in MAEs (Wright and Ma, 2022). For the nonlinear problem, the field of image denoising provides a probabilistic understanding of the unique properties induced in this denoising task (Bioucas-Dias and Figueiredo, 2009; Milanfar and Delbracio, 2024).

CHAPTER 4

A PICTURE OF THE SPACE OF TASKS

In the first chapter, we used learning theory to understand how to optimally train on multiple tasks. The learning theoretic definitions of task-relatedness are too strong and not usable in practice — the theory only applies in the asymptotic regime of $m \to \infty$. The bounds are often vacuous because they are extremely pessimistic in their predictions since they capture all possible distributions over the inputs and for all tasks realizable by the hypothesis class.

In this chapter, we adopt the view that in order to build useful theory, we must build theory that only applies to ”typical tasks”. But before we can build such a theory, we must understand what typical tasks look like. Toward this end, this chapter discusses two results that attempt to characterize the space of typical learnable tasks. We use an atypical set of tools from information theory, signal processing, and information geometry to describe the manifold of tasks and to characterize redundancy in typical tasks.

4.1 Training trajectories of typical tasks lie in a low-dimensional Manifold

In this section, we use techniques from computational information geoemtry to understand the representations learned by deep networks when they are trained on different tasks and using different methods. While algorithms for supervised, meta-, semi-supervised and contrastive learning are very different from each other, they must be exploiting some structure in the space of tasks considering that all of these methods work incredibly well. The goal is to uncover this structure and shed light on *why* these existing algorithms are successful. The tools help us shed light on the following phenomena: (1) the manifold of probabilistic models trained on different tasks using different representation learning methods is effectively low-dimensional; (2) supervised learning on one task results in a surprising amount of progress even on seemingly dissimilar tasks; progress on other tasks is larger if the training task has diverse classes; (3) the structure of the space of tasks indicated by our analysis is consistent with parts of the Wordnet phylogenetic tree; (4) episodic meta-learning algorithms and supervised learning traverse different trajectories during training but they fit similar models eventually; (5) contrastive and semi-supervised learning methods traverse trajectories similar to those of supervised learning.

4.1.1 Methods

The key idea is to think of a deep network with weights w trained on a task as a probabilistic model $P_w(\vec{y})$ where $\vec{y} = (y_1, \ldots, y_N)$ denotes any sequence of outputs (each $y_n \in \{1, \ldots, C\}$ classes) on N independent and identically distributed samples. We instantiate the technical machinery of information geometry using this NC-dimensional object to study different probabilistic models fitted to the task irrespective of which representation learning algorithm, e.g., supervised learning, meta-learning, etc., or what neural architecture was used to fit the probabilistic model. This construction circumvents the enormous diversity of algorithms, architectures with different feature spaces and training methods across these different sub-fields and provides us with a single space to study these models in — the prediction space of the model. Some of these tools are developed in more detail in Mao et al. (2024). These tools are used to visualize these very high-dimensional objects, to compute geodesics on such manifolds, to interpolate checkpoints along training trajectories into

continuous curves, and to map models trained on different tasks into a unique prediction space.

Modeling the task A task P is a joint distribution on inputs $x \in \mathbb{R}^d$ and outputs $y \in \{1, \ldots, C\}$ corresponding to C classes. Suppose we have N independent and identically distributed samples $\left\{(x_n, y_n^*)\right\}_{n=1}^N$ from P. Let $\vec{y} = (y_1, \ldots, y_N)$ denote any sequence of outputs on these N samples and $\vec{y}^*$ denote the sequence of ground-truth labels. We can model the task as

$$P_w(\vec{y}) = \prod_{n=1}^{N} p_w^n(y_n) \tag{4.1}$$

where w are the parameters of the model and we have used the shorthand $p_w^n(y_n) \equiv p_w(y_n \mid x_n)$. Let "truth" or $P_* \equiv P(\vec{y}^*)$ denote the true probability distribution which corresponds to the ground-truth labels. Let "ignorance" or P_0 denote the probability distribution that corresponds to $p^n(y) = 1/C$ for all n and all $y \in \{1, \ldots, C\}$.

Bhattacharyya distance Given two models P_u and P_v parameterized by weights u and v respectively, the Bhattacharyya distance (Bhattacharyya, 1946) between them averaged over samples can be written as

$$\begin{aligned} \mathrm{d_B}(P_u, P_v) &:= -N^{-1} \log \sum_{\vec{y}} \prod_n \sqrt{p_u(y_n)\, p_v(y_n)} \\ &= -N^{-1} \sum_n \log \sum_c \sqrt{p_u^n(c)\, p_v^n(c)}. \end{aligned} \tag{4.2}$$

Our model eq. (4.1) involves a product over the probabilities of N samples. Many distances, e.g., the Hellinger distance $2\left(1 - \prod_n \sum_c \sqrt{p_u^n(c)\, p_v^n(c)}\right)$, saturate for large N, this is because two random high-dimensional vectors are nearly orthogonal. This makes it difficult to use such distances to understand high-dimensional probabilistic models. The Bhattacharyya distance is well-behaved for large N due to the logarithm (Quinn et al., 2019b; Teoh et al., 2020), and that is why it is well suited to our problem.

Remark 23 (Models with different intermediate representations can have zero Bhattacharyya distance)**.** *Two models can have different internal representations and yet define identical probabilistic models. For example, a representation and a rotated version of the same representation can define identical probabilistic models if this rotation is undone before the output. The Bhattacharyya distance eq.* (4.2) *only depends on the output probabilities and would be zero if the probabilistic models are identical. Focusing the theory on the proba-*

bilistic model that makes the predictions as opposed to the feature space therefore allows us to capture many symmetries in the prediction space.

Distances between trajectories of probabilistic models Consider a trajectory $(w(k))_{k=0,\dots,T}$ that records the weights after T updates of the optimization algorithm, e.g., stochastic gradient descent. This trajectory corresponds to a trajectory of probabilistic models $\tilde{\tau}_w = (P_{w(k)})_{k=0,\dots,T}$. We are interested in calculating distances between such training trajectories. First, consider $\tilde{\tau}_u = (u(0), u(1), u(2), \dots, u(T))$ and another trajectory $\tilde{\tau}_v \equiv (u(0), u(2), u(4), \dots, u(T), u(T), \dots, u(T))$ which trains twice as fast but to the same end point. If we define the distance between these trajectories as, say, $\sum_k \mathrm{d_B}(P_{u(k)}, P_{v(k)})$, then the distance between $\tilde{\tau}_u$ and $\tilde{\tau}_v$ will be non-zero—even if they are fundamentally the same. This issue is more pronounced when we calculate distances between training trajectories of different tasks. It arises because we are recording each trajectory using a different time coordinate, namely its own training progress.

To compare two trajectories correctly, we need a notion of time that can allow us to uniquely index any trajectory. The geodesic between the start point P_0 and the true distribution P_* is a natural candidate for this purpose since it is unique. Geodesics are locally length-minimizing curves in a metric space. For the product manifold in eq. (4.1), we can obtain a closed-form formula for the geodesic by noticing that for each sample, the vector $\left(\sqrt{p_u^n(c)}\right)_{c=1,\dots,C}$ lies on a C-dimensional unit sphere. The geodesic connecting two models P_u and P_v under the Fisher information metric which is induced by the Bhattacharyya distance is just the great circle on the sphere (Ito and Dechant, 2020, Eq. 47):

$$\sqrt{P_{u,v}^{\lambda}} = \prod_{n=1}^{N} \left(\frac{\sin\left((1-\lambda)\mathrm{d}_{\mathrm{G}}^n\right)}{\sin\left(\mathrm{d}_{\mathrm{G}}^n\right)} \sqrt{p_u^n} + \frac{\sin\left(\lambda \mathrm{d}_{\mathrm{G}}^n\right)}{\sin\left(\mathrm{d}_{\mathrm{G}}^n\right)} \sqrt{p_v^n} \right), \tag{4.3}$$

where $\lambda \in [0,1]$ and $\mathrm{d}_{\mathrm{G}}^n = \cos^{-1}\left(\sum_c \sqrt{p_u^n(c)}\sqrt{p_v^n(c)}\right)$ is one half of the great-circle distance between $p_u^n(\cdot)$ and $p_v^n(\cdot)$. Any probabilistic model P_w on a trajectory $\tilde{\tau}_w$ can now be **re-indexed by a new "time" that we call "progress"**:

$$[0,1] \ni t_w = \arg \inf_{\lambda \in [0,1]} \mathrm{d_G}(P_w, P_{0,*}^{\lambda}). \tag{4.4}$$

It indicates the distance of P_w to the truth P_* measured in terms of the closest point $P_{0,*}^{t_w}$ on the geodesic to P_w. We solve eq. (4.4) using bisection search (Brent, 1971).

Observe that using the same expression as eq. (4.3), we can also interpolate between two successive recorded points $P_{w(k)}$ and $P_{w(k+1)}$ of a trajectory by calculating $P^{\lambda}_{w(k),w(k+1)}$ for different values of $\lambda \in [0, 1]$. This is useful because different networks train with very different speeds on different tasks, especially in early stages of training. This allows us to effectively convert a sequence of models $\tilde{\tau}_w = (P_{w(k)})_{k=0,\dots,T}$ into a continuous curve $\tau_w = (P_{w(t)})_{t\in[0,1]}$. We calculate the distance between continuous curves τ_u, τ_v as

$$\mathrm{d}_{\mathrm{traj}}(\tau_u, \tau_v) = \int_0^1 \mathrm{d}_{\mathrm{B}}(P_{u(t)}, P_{v(t)})\,\mathrm{d}t\,; \tag{4.5}$$

which is approximated using a uniform grid on $[0, 1]$.

Riemann length of a trajectory Divergences like the Bhattacharyya distance or the Kullback-Leibler (KL) divergence (which is the cross-entropy loss up to a constant) can be used to define a Riemannian structure in the space of probabilistic models (Amari, 2016). The distance between two infinitesimally different models P_w and $P_{w+\mathrm{d}w}$ is

$$\mathrm{d}s^2 = 4\mathrm{d}_{\mathrm{B}}(P_w, P_{w+\mathrm{d}w}) = \left\langle \mathrm{d}w\,, g(w)\,\mathrm{d}w \right\rangle + o(\|\mathrm{d}w\|^2),$$

where $g(w) = N^{-1}\sum_{\vec{y}}(P_w)^{-1}\partial^2 P_w$ is the Fisher Information Matrix (Quinn, 2019, Section A.3). This Fisher Information Matrix (FIM) is therefore the metric of the space of the probability distributions and weights w play the role of the coordinates in this space. Up to a scalar factor, the Bhattacharyya distance and the KL-divergence induce the same FIM. The Riemann length of a trajectory τ_w is the integral of these infinitesimal lengths:

$$\mathrm{Length}(\tau_w) = 2\int_0^1 \sqrt{\mathrm{d}_{\mathrm{B}}(P_{w(t)}, P_{w(t+\mathrm{d}t)})}; \tag{4.6}$$

it is equal to the integral of FIM-weighted incremental distance traveled in the weight space. Observe that we do not need the FIM to calculate the length. We can think of the length of a trajectory taken by a model to reach the solution P_* compared to the length of the geodesic as a measure of the inefficiency of the training procedure since the geodesic is the curve with the shortest length. This inefficiency can arise because: (a) not all probability distributions along the geodesic can be parametrized by our model class (approximation error), and (b) the training process may take steps that are misaligned with the geodesic (e.g., due to the loss function, mini-batch updates, supervised vs. some other form of representation learning, etc.).

Mapping a model trained on one task to another task using "imprinting" We will consider different tasks $\{P^k\}_{k=1,\dots,}$ with the same input domain but possibly different number of classes C^k. Given a model P^1_w parametrized by weights w for task P^1, we are interested in evaluating its learned representation on another task, say, P^2. Let $w = (w_1, w_2)$ be the weights for the backbone and the classifier respectively. The logits are $\mathbb{R}^{C^1} \ni {w_2}^\top \varphi(x; w_1)$ corresponding to an input x and features of the penultimate layer $\varphi(x; w_1)$. The network's output $p_w(c \mid x_n)$ for $c = 1, \dots, C^1$ is computed using a softmax applied to the logits. If we have learned w from one task P^1, then we can re-initialize each row of the classifier weights $(w_2)'_c$ for $c = 1, \dots, C^2$ to maximize the cosine similarity with the average feature of samples from task P^2 with ground-truth class c:

$$(w_2)'_c = h/\|h\|_2 \quad \text{where } h = \textstyle\sum_{\{x : y^*_x = c\}} \varphi(x; w_1). \tag{4.7}$$

The new network $w = (w_1, {w_2}')$ can be used to make predictions on P^2. Using imprinting, we can map a trajectory τ^1_w of a network being trained on P^1 to another task P^2 by mapping each point along the trajectory; let us denote this mapped trajectory by $\tau^{1\to 2}_w$.

Remark 24 (Imprinting versus training the final layer or probing)**.** *There are many ways of performing such a mapping, e.g., one could fine-tune the weights using data from P^2, linear probing (Shi et al., 2016), etc. The technique described above is known as "imprinting" (Hu et al., 2015; Qi et al., 2018; Dhillon et al., 2020). In this work, we will be mapping thousands of models across different trajectories to other tasks. Training the final layer, or a new classifier, for all these models is cumbersome and imprinting provides a simple way around this issue. Note that imprinting is not equivalent to training the classifier w_2 (with backbone w_1 fixed) using samples from the other task but we found that imprinted weights work well in practice.*

How to choose an appropriate task to map different models to? Consider the training trajectory τ^1_u of a model being trained on P^1 and another trajectory τ^2_v of a model being trained on P^2. Using eq. (4.7), we can map these trajectories to the other task to get $\tau^{1\to 2}_u$ and $\tau^{2\to 1}_v$. This allows us to calculate, for instance, $\mathrm{d}_{\mathrm{traj}}(\tau^{1\to 2}_u, \tau^2_v)$ using eq. (4.5) which is the distance of the trajectory of the model trained on P^1 and then mapped to P^2 with respect to the trajectory of a model trained on task P^2. If the two learning tasks P^1 and P^2 are very different, (e.g., Animals in CIFAR-10 and Vehicles in CIFAR-10), then this distance will be large.

Quantities like $\mathrm{d}_{\mathrm{traj}}(\tau^{1\to 2}_u, \tau^2_v)$ or $\mathrm{d}_{\mathrm{traj}}(\tau^{2\to 1}_v, \tau^1_u)$ are reasonable candidates to study similarities between tasks

P^1 and P^2, but they are not equal to one another. We are also interested in doing such calculations with models trained on many different tasks, and mapping them to each other will lead to an explosion of quantities. To circumvent this, we map to a unique task whose output space is the union of the output spaces of the individual tasks, e.g., to study P^1 (Animals) and P^2 (Vehicles), we will map both trajectories to P^{U} which is all of CIFAR-10. We will use

$$\mathrm{d}_{\mathrm{traj}}(\tau_u^{1\to \mathrm{U}}, \tau_v^{2\to \mathrm{U}}) \tag{4.8}$$

as the distance between trajectories trained on P^1 and P^2.

Visualizing a high-dimensional probabilistic model in lower-dimensions We use a visualization technique called intensive principal component analysis (InPCA) (Quinn et al., 2019b) that embeds a probabilistic model into a lower-dimensional space. For m probability distributions, consider a matrix $D \in \mathbb{R}^{m\times m}$ with entries $D_{uv} = \mathrm{d}_{\mathrm{B}}(P_u, P_v)$ and

$$W = -LDL/2 \tag{4.9}$$

where $L_{ij} = \delta_{ij} - 1/m$ is the centering matrix. An eigen-decomposition of $W = U\Sigma U^\top$ where the eigenvalues are sorted in descending order of their magnitudes $|\Sigma_{00}| \geq |\Sigma_{11}| \geq \ldots$ allows us to compute the embedding of these m probability distributions into an m-dimensional space as $\mathbb{R}^{m\times m} \ni X = U\sqrt{\Sigma}$. Unlike standard PCA where eigenvalues are non-negative, eigenvalues of InPCA can be both positive and negative, i.e., the lower-dimensional space is a Minkowski space (Quinn et al., 2019b). This allows the InPCA embedding to be an isometry, i.e., pairwise distances are preserved:

$$\textstyle\sum_{i=1}^{m}(X_u^i - X_v^i)^2 = \mathrm{d}_{\mathrm{B}}(P_u, P_v) \geq 0 \tag{4.10}$$

for embeddings X_u, X_v of two distributions P_u, P_v. We can measure how well pairwise distances are preserved by a k-dimensional sub-space using the "explained stress" χ_k (Cox and Cox, 2000):

$$\chi_k = 1 - \frac{\left\| W - \sum_{i=1}^{k} \Sigma_{ii}\, U_i U_i^\top \right\|_{\mathrm{F}}}{\|W\|_{\mathrm{F}}} = 1 - \sqrt{\frac{\sum_{i=k+1}^{m} \Sigma_{ii}^2}{\sum_{i=1}^{m} \Sigma_{ii}^2}}. \tag{4.11}$$

Just like standard PCA, if we preserve all the eigenvectors (i.e., $k = m$), then eq. (4.10) holds exactly and $\chi_k = 1$. But if we use fewer eigenvectors then pairwise distances can be distorted.

4.1.2 Results

We next describe our findings using the theoretical ideas developed in the previous section. We present a broad range of evidence and analysis using a large number of representation learning techniques, multiple neural architectures and a large number of different image-classification tasks created from the CIFAR-10 and ImageNet datasets. Experiments in this work required about 30,000 GPU-hours.

Remark 25. *All the analysis (except figs. 4.4 and 4.5) was conducted using the test data. All models were trained using the training data, but all mapped models, distances between trajectories, quantitative evaluation of progress and InPCA embeddings were computed using the test dataset. The reason for this is that we would like to study the geometry of tasks as evidenced by samples that were not a part of training. To emphasize, we do not develop any new algorithms for learning. Therefore using the test data to quantify relationships between tasks is reasonable; see similar motivations in Kaplun et al. (2022) or Ilyas et al. (2022) among others. Our findings remain valid when training data is used for analysis; this is because in most of our experiments, a representation is trained on one task but makes predictions on a completely new task after mapping.*

Result 1: The manifold of models trained on different tasks, and using different representation learning methods, is effectively low-dimensional We trained multiple models on 6 different sub-tasks of ImageNet (from 5 random initializations each) to study the dimensionality of the manifold of probabilistic models along the training trajectories (100 points equidistant in progress eq. (4.4)) after mapping all models to all ImageNet classes ($\sim 10^8$ dimensions). We use the explained stress, to measure if the distances are preserved by the first k dimensions of the embedding of the models. The first 3 dimensions of InPCA (fig. 4.1a) preserve 80.02% of the explained stress (fig. 4.1b shows more dimensions). This is therefore a remarkably low-dimensional manifold. It is not exactly low-dimensional because the explained stress is not 100%, but it is an effectively low-dimensional manifold. This also indicates that the individual manifolds of models trained on one task are low-dimensional, even if they start from different random initializations in the weight space. Such low-dimensional manifolds are seen in *all* our experiments, irrespective of the specific method used for

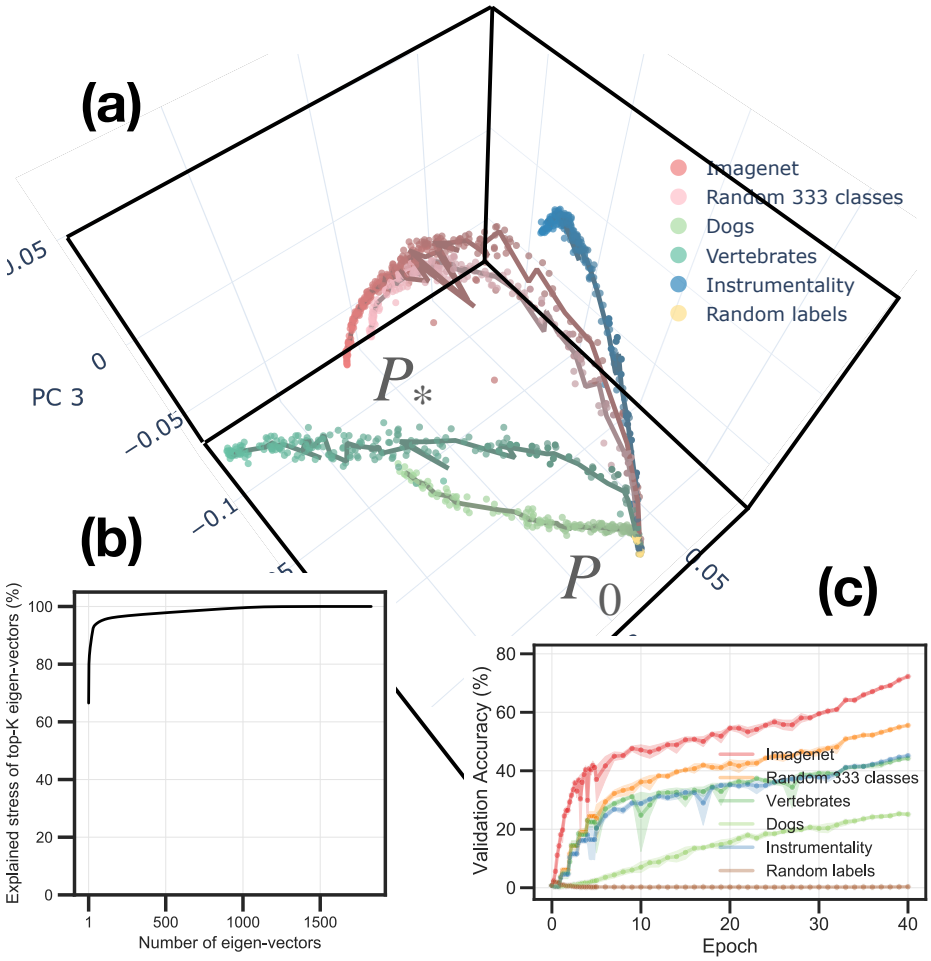


Figure 4.1: (a) Visualization of training trajectories of models trained on 6 tasks from ImageNet. Each point is one network, bold lines connect points along the average trajectory of each task (across 5 random weight initializations). Trajectories move towards the truth P_*, which corresponds to the ground-truth labels. Training on one task makes a remarkable amount of progress on unseen, seemingly dissimilar, classes. Trajectories of models trained on a random set of 333 classes are similar to those of the entire ImageNet. Some classes (Instrumentality) are closer to this trajectory while others such as Vertebrates and Dogs are farther away. Dogs is a semantic subset of Vertebrates; it splits at the beginning but seems to eventually reach a similar representation as one of the intermediate points of Vertebrates.

(b) Percentage explained stress eq. (4.11) captured by subspace spanned by the top k InPCA eigenvectors.

(c) Validation accuracy on different tasks vs. epochs.

representation learning, namely, supervised, transfer (fine-tuning), meta, semi-supervised and contrastive learning.

Remark 26 (A detailed description of how we plot trajectories of representations)**.** *We provide a non-mathematical description of how the theory was used to draw fig. 4.1a below. We train 5 different networks (random seeds for initialization) for each of the 6 tasks, and record 61 model checkpoints during training;*

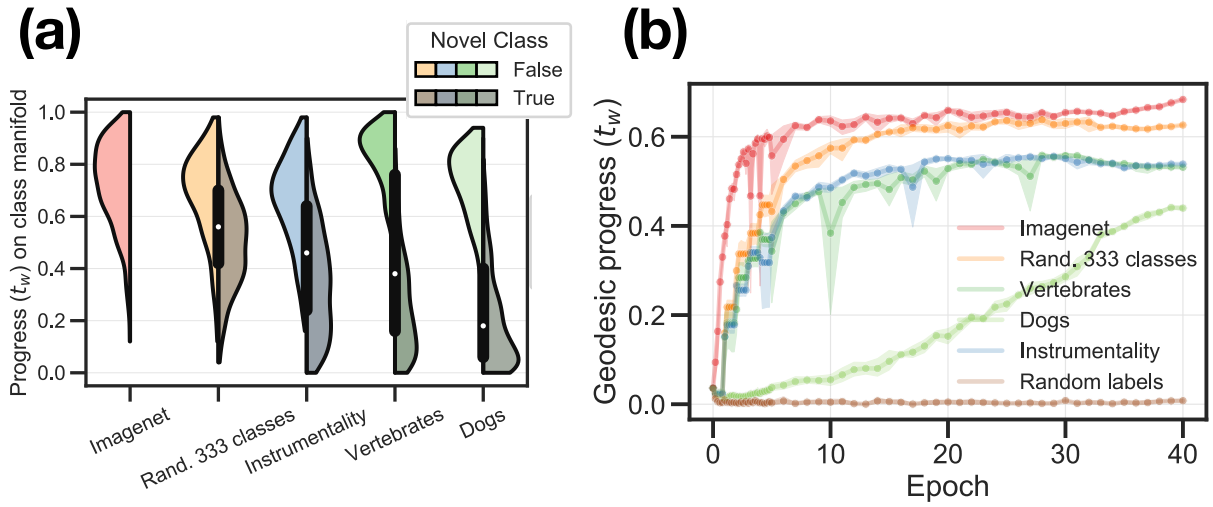


Figure 4.2: **(a)** Progress made by each model on classes seen during training (left half, lighter shade) and on novel classes (right half, darker shade). We compute t_w^c which is the progress t_w of images restricted to a single class c. This quantity t_w^c measures the quality of the representation for class c. Violin plots denote the distribution of t_w^c indicate that we make more progress on classes seen during training. If the model sees a larger diversity of classes (like with random 333 classes), more progress is made on the novel classes. Surprisingly, even if we train on just the "Dogs", we make some progress on novel classes.
(b) Progress t_w eq. (4.4) on the Y-axis against the number of epochs of training on the X-axis. The progress t_w increases with more epochs of training—all models make non-trivial progress towards the truth P_* ($t_w = 1$). Even if we train on only Dogs (118 classes) we make progress on the entire ImageNet.

this gives 1830 checkpoints for this experiment. We re-index all checkpoints to calculate their progress using eqs. (4.3) *and* (4.4)*. We then interpolate between each consecutive pair of the 61 checkpoints along each trajectory using eq.* (4.3)*. The training trajectory can now be sampled at any progress* $t_w \in [0, 1]$*. We next calculate the "average trajectory" of the 5 networks (random seeds) of each task by averaging the output probabilities in eq.* (4.1) *at a fixed value of* t_w*; 100 different values of* t_w *spread uniformly between* $[0, 1]$ *are chosen. These 100 points along the average trajectory of each of the 6 tasks are also embedded together with the 1830 checkpoints (i.e.,* $m = 2430$ *in eq.* (4.9))*. fig.* 4.1*a plots the top three dimensions obtained from InPCA. To clarify, the explained stress of the top 2430 dimensions would be exactly 100%.*

Result 2: Supervised learning on one task results in a surprising amount of progress on seemingly dissimilar tasks. Progress on other tasks is larger if the training task has diverse classes. We studied the progress t_w eq. (4.4) made by models (fig. 4.2b) trained on tasks from Result 1. Training on the task "Dogs" makes non-trivial progress on other tasks, even seemingly dissimilar ones like "Instruments" which contains vehicles, devices and clothing. In fact, it makes a remarkable amount of progress on the *entire*

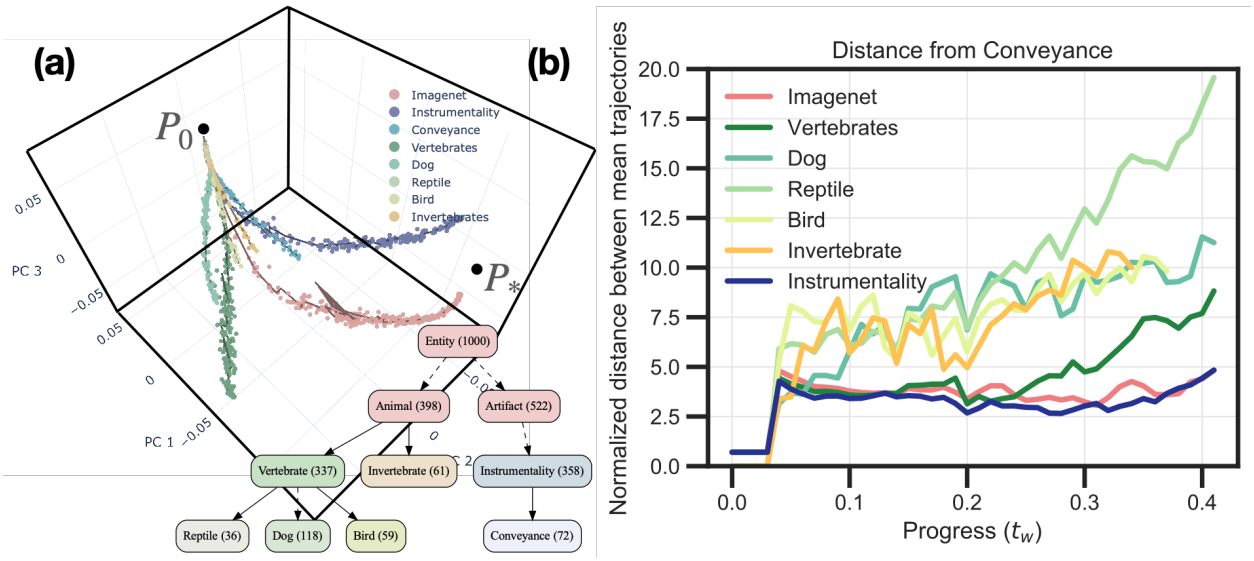


Figure 4.3: **(a)** Trajectories of models trained on different phyla of Wordnet (inset). The model manifold is again effectively low-dimensional (78.72% explained stress in 3 dimensions).

(b) We analyze the trajectories in fig. 4.3(a) and obtain a quantitative description of how trajectories of different tasks diverge from each other during training; the procedure is explained in in remark 27. The plot depicts the Bhattacharyya distance between the mean trajectories (over random initializations) on different tasks, and the mean trajectory of Conveyance. This distance is normalized by the average of the tube radii (maximum distance of one of the 5 trajectories from the mean, computed at each progress) of the two trajectories. Such quantities allow us to make precise statements about the differences between representations and show some very surprising conclusions. Trajectories of tasks that are nearby in Wordnet are also nearby in terms of their learned representations. Further, trajectories of ImageNet (pink) are closer to Conveyance (as expected), but those of Vertebrates (red) are equally far away for more than 60% ($t_w \approx 0.25$) of the progress. In other words, training on Vertebrates (reptiles, dog, bird) makes a remarkable progress on Conveyance (cars, planes).

ImageNet, about 63.38% of the progress of a model trained directly on ImageNet. Progress is larger for larger phyla of ImageNet (Vertebrates and Instruments). But what is surprising is that if we train on a random subset of 333 classes (a third of ImageNet), then the progress on the entire ImageNet is very large (92%). This points to a strong shared structure among classes even for large datasets such as ImageNet. Note that this *does not* mean that tasks such as Vertebrates and Instruments are similar to each other. Even if training trajectories are similar for a while, they do bifurcate eventually and the final models are indeed different (see fig. 4.3b and remark 27 on how to interpret it).

In fig. 4.2a, we studied the projections of models trained on one task onto the geodesics of unseen classes calculated using eq. (4.3) evaluated at the progress t_w eq. (4.4)). We find that a model trained on the entire ImageNet makes uneven progress on the various classes (but about 80% progress across them, progress is highly correlated with test error of different classes). Models trained on the 6 individual tasks also make

progress on other unseen classes. As before, training on Instruments, Vertebrates, Dogs makes smaller progress on unseen classes compared to training on a random subset of 333 classes. This is geometric evidence that the more diverse the training dataset, the better the generalization to *unseen* classes/tasks; this phenomenon has been widely noticed and utilized to train models on multiple tasks, as we discuss further in Result 4.

Result 3: The structure of the space of tasks indicated by our visualization technique is consistent with parts of the Wordnet phylogenetic tree. To obtain a more fine-grained characterization of how the geometry in the space of learnable tasks reflects the semantics of these tasks, we selected two particular phyla of ImageNet (Animals, Artifacts) and created sub-tasks using classes that belong to these phyla (fig. 4.3a). Trajectories of models trained on Instruments and Conveyance are closer together than those of Animals. Within the Animals phylum, trajectories of Vertebrates (Dog, Reptile, Bird) are closer together than those of Invertebrates (fig. 4.3b for quantitative metrics). Effectively, we can recover a part of the phylogenetic tree of Wordnet using our training trajectories. We speculate that this may point to some shared structure between visual features of images and natural language-based semantics of the corresponding categories which was used to create Wordnet (Miller, 1995) of the corresponding categories. Such alignment with a natural notion of relatedness also demonstrates the soundness and effectiveness of our technical machinery.

Remark 27 (Building a precise and quantitative characterization of trajectories of representations)**.** *The precise way to understand statements like those in Result 3 is using the quantitative analysis reported in fig. 4.3b. To expand upon the caption, the X-axis of the plot is progress. For multiple models (5 random seeds) trained on two tasks (say Conveyance and Dogs), we have calculated the mean (across random seeds) of the interpolated trajectories at different progress. At each specific progress, we have plotted the distance between the mean model trained on Conveyance (say task 1) and Dogs (say task 2) divided by the average tube radii (which is the maximum of the distance of the model corresponding to one seed from the mean):*

$$2d_B(\tau^{1\to U}_{mean}, \tau^{2\to U}_{mean})/\textstyle\sum_{k=1,2} \max_a [d_B(\tau^{k\to U}_a, \tau^{k\to U}_{mean})].$$

The is a measure of how far away the trajectories of these two models are. If it is less than 1, then the "tubes" corresponding to models trained on tasks 1 and task 2 intersect.

Let us emphasize that we have performed such analyses for all experiments; while the InPCA embedding gives an easy-to-understand visual description of these results for high-dimensional probabilistic models, the information geometric techniques developed enable us to make these descriptions precise and quantitative.

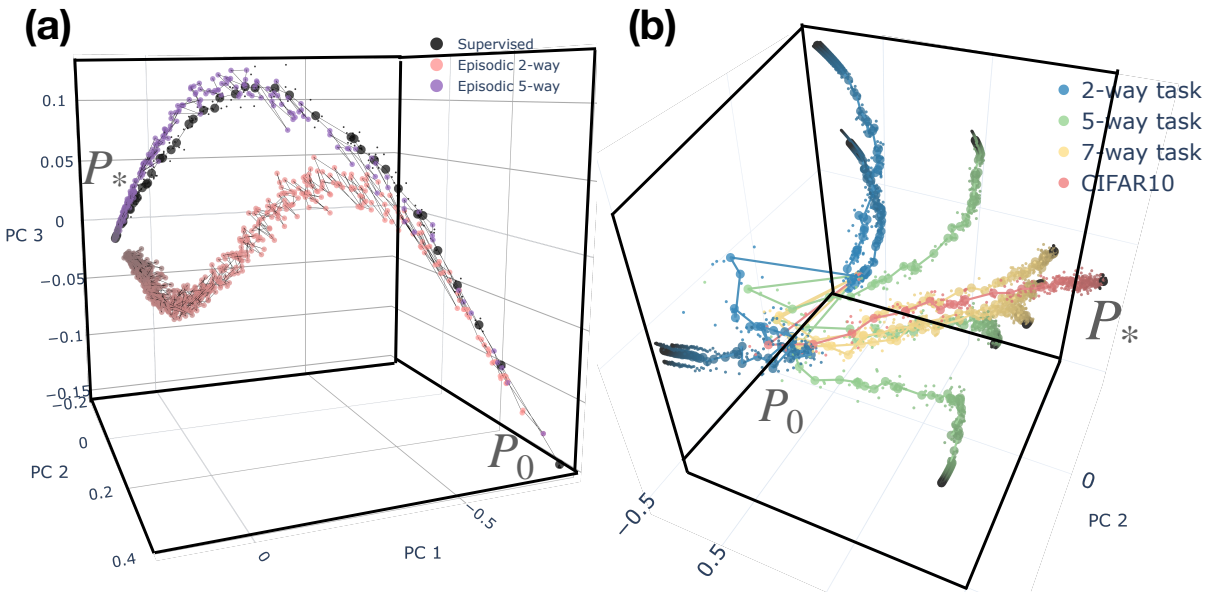


Figure 4.4: **(a)** Training trajectories for supervised learning (black), 2-way (pink) and 5-way episodic meta-learning (purple). Trajectories of 5-way meta-learning are very similar to those of supervised learning and eventually reach very similar models and high test accuracy. In contrast, 2-way meta-learning has a much longer trajectory (about 40× longer in Riemann length than black) and does not reach a good test accuracy (on all 10 CIFAR-10 classes). Representations are similar during early parts of training even if these are quite different learning mechanisms.
(b) Trajectories of 2-way (blue), 5-way (green), 7-way (yellow) tasks trained using cross-entropy loss compared to supervised learning (red). For large "way", trajectories are similar to supervised learning but they quickly deviate from the red trajectories for small ways.

Result 4: Episodic meta-learning algorithms traverse very different trajectories during training but they fit a similar model eventually. Meta-learning methods build a representation which can be adapted to a new task (Thrun and Pratt, 1998). We studied a common variant, the so-called episodic training methods (Bengio et al., 1992), in the context of few-shot learning methods (Vinyals et al., 2016). In these methods, each mini-batch consists of samples from C^w out of C classes (called "way") split into two parts: a "support set" D_s of s samples/class (called "shot"), and a "query set" D_q of q samples/class. Typical methods, say prototypical networks of Snell et al. (2017), implement a clustering loss on features of the query samples using averaged features of the support samples $\varphi_c = s^{-1} \sum_{\{x \in D_s, y^*(x)=c\}} \varphi(x; w_1)$ for all $c = 1, \ldots, C^w$ as

the cluster centroids. If features φ lie on an ℓ_2 ball of radius 1, then doing so is akin to maximizing the cosine similarity between cluster centroids and features of query samples. The same clustering loss with the learned backbone w_1 is used to predict on unseen classes (using "few" support samples to compute centroids) at test time.

To understand the representations learned by episodic meta-learning methods, we compared trajectories of episodic meta-learning to the trajectory taken by supervised learning in fig. 4.4. Supervised learning uses the cross-entropy loss over all the C classes while episodic meta-learning optimizes a loss that considers all k-way classification tasks (where k is typically smaller than C), its objective differs from that used for supervised learning. Since the two objectives are different, it comes as a surprise that both arrive at the same solution; see fig. 4.4a,b for distances between trajectories. But the Riemann trajectory length of episodic training is about 40× longer than that of supervised learning. It is worth noting that the explained stress is only 40.96% in fig. 4.4a because of larger fluctuations for episodic learning in other directions. Therefore, episodic meta-learning has a qualitatively different training trajectory in the prediction space than supervised learning. The implications of this are consistent with recent literature which has noticed that the performance of few-shot learning methods using supervised learning (followed by fine-tuning) is comparable to, or better than, episodic meta-learning (Dhillon et al., 2020; Kolesnikov et al., 2020; Fakoor et al., 2020). Indeed, a supervised learned representation also minimizes the clustering loss.

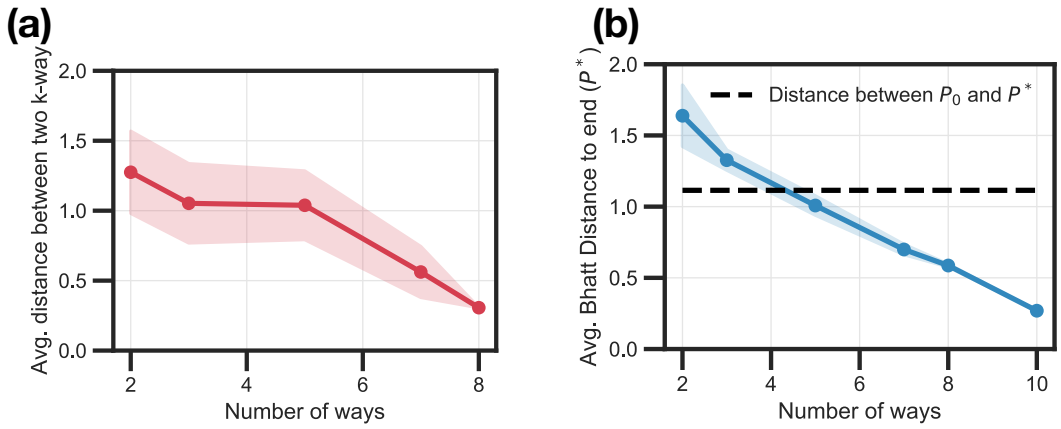


Figure 4.5: **(a)** Average distance between two k-way meta-learning trajectories decreases with k, this is a geometric evidence of the variance of predictions of learned representations. **(b)** Training with a small way leads to models that predict poorly on test data (large distances from truth). These embeddings were calculated using the training dataset. The rationale being that we wanted to show how different meta-learning and supervised learning are during training.

In order to understand why few-shot accuracy of episodic training is better when k is larger for k-way meta-learning (Gidaris and Komodakis, 2018), we trained models on different 2-way 5-way and 7-way tasks using the cross-entropy loss (fig. 4.4b). We find that the radius of the tube that encapsulates the models of 2-way tasks around their mean trajectory is very large, almost as large as the total length of the trajectory, i.e., different models trained with a small way tasks traverse very different trajectories. Tube radius decreases as the way increases (fig. 4.5a). Further, the distance of models from the truth P_* (which is close to the end point of the supervised learning model) is higher for a small way (fig. 4.5b). This is geometric evidence of the widely used empirical practice of using a large way in episodic meta-learning. Observe in fig. 4.5b that as the way increases, the trajectory becomes more and more similar to that of supervised learning.

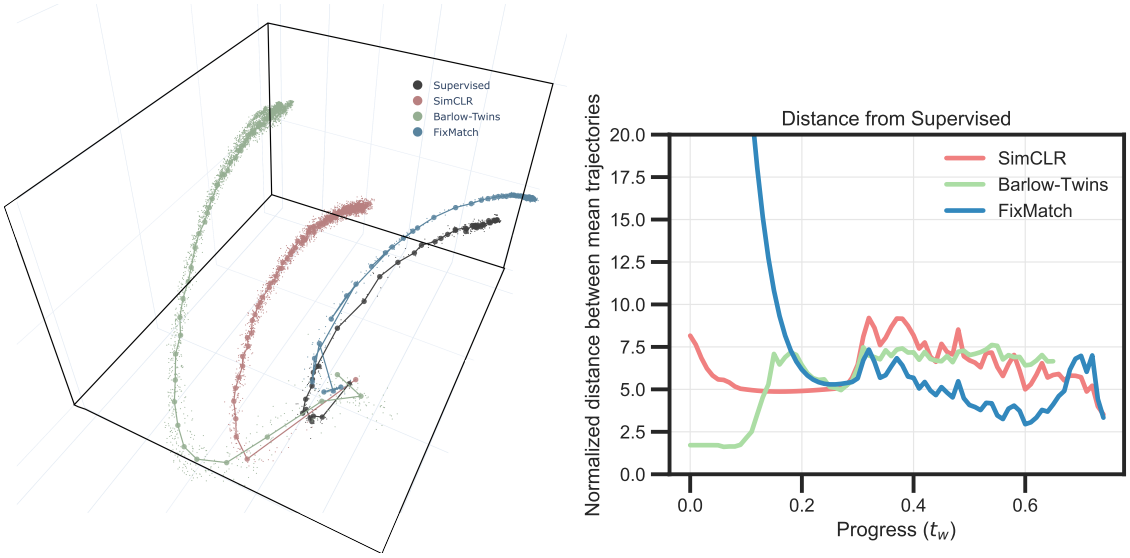


Figure 4.6: We consider 4 methods for training on CIFAR10: supervised learning, SimCLR (Chen et al., 2020a), Barlow-twins (Zbontar et al., 2021) and Fixmatch (Sohn et al., 2020). Fixmatch has access to 2500 labeled samples and 47500 unlabeled samples. SimCLR and Barlow-twins use 50,000 unlabeled samples for training.
(a) We plot the trajectories for supervised, semi-supervised and contrastive learning. The trajectory of semi-supervised learning (Fixmatch) eventually resembles supervised learning in comparison to contrastive learning methods. All methods result in remarkably similar trajectories although some of these methods are trained using only unlabeled data.
(b) Normalized distance of trajectories corresponding to contrastive and semi-supervised learning to the trajectory of supervised learning. Semi-supervised learning (Fixmatch) deviates considerably from the other methods at the beginning. We speculate that this is because the trajectory of Fixmatch is influenced by the 2500 labeled samples. As, training progresses, Fixmatch becomes increasingly similar to supervised learning as evidenced by the dip in the blue line for larger values of progress (t_w).

Result 5: Contrastive and semi-supervised learning methods traverse trajectories similar to those of supervised learning. Contrastive learning (Becker and Hinton, 1992) learns representations without using ground-truth labels (Gutmann and Hyvärinen, 2010; Chen et al., 2020a). It has been extremely effective for self-supervised learning (Doersch and Zisserman, 2017; Kolesnikov et al., 2019), e.g., prediction accuracy

with 1–10% labeled data is close to that of supervised learning using all data (Chen et al., 2020b). Semi-supervised methods (Berthelot et al., 2019b; Sohn et al., 2020) learn representations when ground-truth labels are available for only a small fraction of the data (0.1–1%). These methods achieve a prediction accuracy within 5% of the accuracy achieved through supervised learning. We compared representations learned using contrastive and semi-supervised learning with those from supervised learning to understand why these methods are so effective.

Consider a task P and a set of augmentations G such as cropping, random-resizing, blurring, color, contrast, brightness distortion etc.). Given inputs images x from P, contrastive learning forces the representation $\varphi(g(x); w_1)$ and $\varphi(g'(x); w_1)$ (shortened to $\varphi(g(x))$ below) of the same input for two different augmentations g, g' to be similar. And forces it to be different from representations of other augmented inputs x' (Zbontar et al., 2021; Bachman et al., 2019; Dosovitskiy et al., 2014). Semi-supervised learning methods have access to both labeled inputs x_l and unlabeled inputs x_u. More recent methods are usually trained to fit the labeled inputs using the cross-entropy loss while enforcing consistent predictions across all augmentations (Tarvainen and Valpola, 2017; Berthelot et al., 2019b) for any unlabeled input.

We compare the representations of semi-supervised learning (Fixmatch (Sohn et al., 2020)), contrastive learning(SimCLR (Chen et al., 2020a), Barlow-twins (Zbontar et al., 2021)) and supervised learning in fig. 4.6. All three trajectories are similar to the trajectory of supervised learning. We find that the trajectory of semi-supervised learning deviates from the supervised learning trajectory initially, but the two are very similar for larger values of progress (t_w). This points to a remarkable ability of semi and self-supervised learning methods to learn representations that are similar to those of supervised learning; it is not just that the accuracy of these methods is similar, they also learn similar probabilistic models.

Result 6: Fine-tuning a pre-trained model on a sub-task does not change the representation much. To understand how models train on multiple tasks, we selected two binary classification sub-tasks of CIFAR-10 (Airplane vs. Automobile, and Bird vs. Cat).

We selected models at different stages of standard supervised learning on CIFAR-10 (i.e., using 10-way output and softmax cross-entropy loss) and fine-tuned each of these models on two sub-tasks (the entire

network is fine-tuned without freezing the backbone). As fig. 4.7 shows, models that were fine-tuned from earlier parts of the trajectory travel a large distance and move away from trajectories of the supervised learned CIFAR-10 models. As we fine-tune later and later models, the distance traveled away from the trajectory is smaller and smaller, i.e., changes in the representation are smaller. For a fully-trained CIFAR-10 model which interpolates the training data, the distance traveled by fine-tuning is very small (the points are almost indistinguishable in the picture); this is because both P^1 and P^2 are subsets of CIFAR-10.

Algorithms for transfer learning train on a source task before fine-tuning the model on the target task. If two tasks share a large part of their training trajectory, then we may start the fine-tuning from many shared intermediate points—there are many such points. If the chosen point is farther along in terms of progress then the efficiency resulting from using the source task is higher because the trajectory required to fit the target task is shorter; such trajectories were used in (Gao and Chaudhari, 2021) to define a distance between tasks. As we saw in Result 2, trajectories of different tasks bifurcate after a shared part. The resultant deviation less for related tasks and more for dissimilar tasks (fig. 4.7a, fig. 4.1a,c). Therefore it is difficult to know *a priori* from which point one should start the fine-tuning from without knowing the manifold of the target task. In particular, our geometric picture indicates that fine-tuning from a fully-trained model can be detrimental to the accuracy on the target task. This has been noticed in a number of places in the transfer learning literature, e.g., Li et al. (2020a), and has also been studied theor etically (Gao and Chaudhari, 2020).

Result 7: Contrastive learning methods trained on different datasets learn similar representations We compared representations learned using contrastive learning with those from supervised learning to understand some aspects of why the former are so effective.

We used SimCLR (Chen et al., 2020a) to perform contrastive learning on images from four sets of classes (airplane-automobile, bird-cat, ship-truck and all of CIFAR-10). We compared the learned representation to that from supervised learning on two tasks (airplane-automobile and all of CIFAR-10) in fig. 4.8. Models trained using contrastive learning on two-class datasets learn very different representations from models trained on the same task but using supervised learning. Models trained using contrastive learning on different datasets learning similar representations (trajectories of all three two-class datasets are very close to each other). This is reasonable because contrastive learning does not use any information from the labels. It

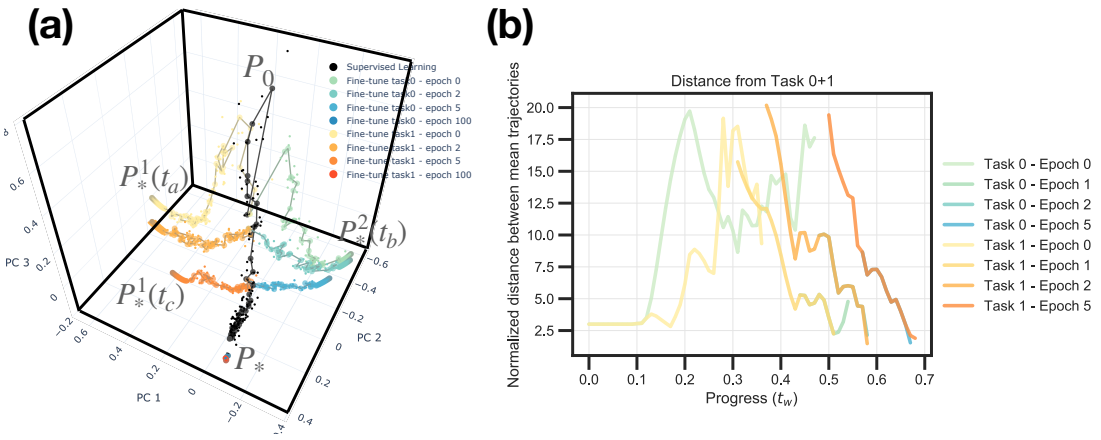


Figure 4.7: **(a)** Fine-tuning trajectories on Airplane vs. Automobile, and Bird vs. Cat sub-tasks of CIFAR-10 (warm and cold hues) pre-trained from different points along the trajectory of supervised learning. If the pretrained model has progressed further towards the truth P_*, then fine-tuning it on a sub-task does not change the representation much. The final trajectory (fine-tuning from epoch 100) is indistinguishable from P_*. **(b)**beginning and determines the efficiency of fine-tuning. Some curves here are not visible because they are overlapping heavily.

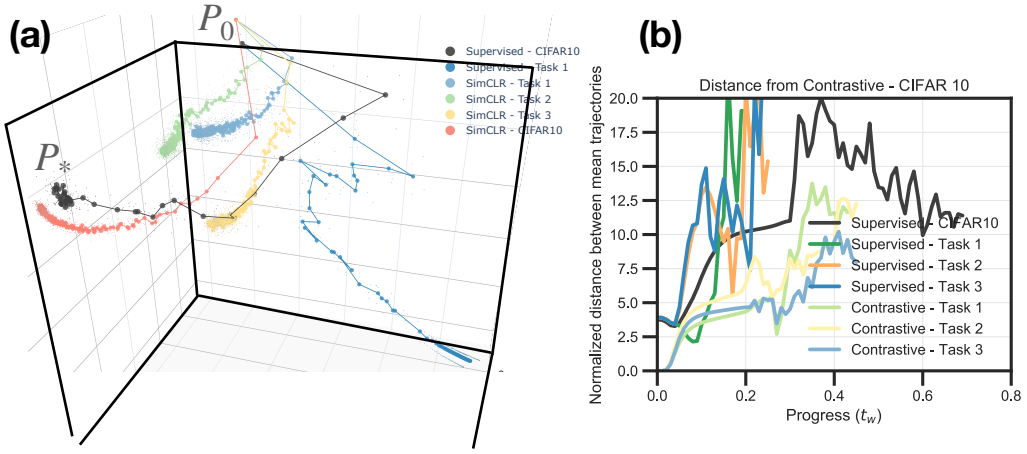


Figure 4.8: **(a)** Trajectories of contrastive learning (SimCLR) on 3 datasets (two classes each) and entire CIFAR-10 compared to those of supervised learning. SimCLR on entire CIFAR-10 learns a similar representation as that of the supervised learned model P_* (which fits the training data perfectly). SimCLR trajectories are close to each other even if different datasets were used to train them. It may seem from the embedding that SimCLR trajectories are similar to that supervised learning, which would be very surprising because the former does not use any labels, but see below.

(b) Bhattacharyya distance between the mean trajectories of all models and the mean trajectory of SimCLR on all CIFAR-10. This distance is normalized by the average of the tube radii (like fig. 4.7b). SimCLR trajectories of two-class datasets are indeed very close to each other (mean distance is $\sim 5\times$ more than their tube radii for about 45% of the way ($t_w \approx 0.2$)). This plot indicates that two-class SimCLR trajectory (light blue) is close to SimCLR on all of CIFAR-10. But two-class supervised learning trajectory (darker blue) is much farther away from SimCLR on all of CIFAR-10.

is surprising however that the trajectory of models from contrastive learning on these two-class datasets is similar to trajectories of models from contrastive learning on the entire CIFAR-10.

Let us elaborate upon this a bit more. We have color-matched the lines in fig. 4.8b with those in fig. 4.8a. The black curve is the trajectory of supervised learning on the entire CIFAR-10; red is the trajectory of SimCLR trained on the entire CIFAR-10. fig. 4.8b compares the distances of trajectories in fig. 4.8a from the red one "contrastive"; this is why there is no red trajectory in fig. 4.8b.

- The first thing to note here is that the black and red trajectories are quite close to each other; the black line in fig. 4.8b is only about 20 times far away from red as compared to their corresponding tube radii.
- Next observe that the trajectory of SimCLR on Task 1 (light blue), SimCLR on Task 2 (green) and SimCLR on Task 3 (yellow) are very similar to each other; this is seen in both fig. 4.8a and in fig. 4.8b.
- Third, they are closer to SimCLR on all of CIFAR-10 than any supervised learning trajectories (this is seen in fig. 4.8b because their curves are below everyone else). Thus, contrastive learning on datasets with different classes learns similar representations.
- The learned representation of two-class SimCLR models is similar to the one obtained using data from all classes (red) (in this experiment this occurs up to about $t_w = 0.4$ progress) but they do not go all the way to the truth (i.e., the end point of black line). This shows the benefit of having data from many classes during contrastive learning.

4.2 Many perception tasks are redundant functions of the input

Suppose we are given a dataset with inputs that are vectors in Euclidean space and outputs denoting each input's ground-truth category. The textbook procedure for modeling this data often begins with principal component analysis (PCA (Hotelling, 1933)). PCA projects inputs onto the "principal" subspace where the variance of the projection is maximal. PCA can be used to reduce the dimensionality, remove the effects of noise, and, as every reader has done in the past, fit a model for predicting the labels using these salient features. The larger the explained variance of PCA, the smaller (hopefully) the information about the labels thrown away by the projection. The smaller the dimension of the principal subspace, the more robust the model is to data variations in the discarded subspace. This is why any data science textbook teaches its readers to identify the "elbow" in a scree plot as the size of the principal subspace (James et al., 2013).

This paper shows that for many perception tasks—from visual recognition, semantic segmentation, optical flow, depth estimation to auditory discrimination—one can accurately predict the output using non-salient features. The principal subspace is most predictive of these tasks. However, the predictive ability of any other subspace is remarkably high. These perception tasks are, therefore, highly redundant functions of their input data. We examine this phenomenon through different lenses, using ideas from signal processing, information theory, and neuroscience. section 4.2.2 discusses our results, where we identify common themes and important differences across these modalities.

4.2.1 Methods

Principal components analysis (PCA), Fourier transform and the wavelet transform. The input to all of our image filters is a discretely-sampled image $x \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ with height d_1, width d_2 and d_3 channels. For PCA, we flatten the image and subtract the mean over all images to form $X \in \mathbb{R}^{d \times n}$ where n is the number of examples and $d = d_1 d_2 d_3$. We then form the sample covariance matrix given by $\Sigma = XX^\top/n$. PCA computes the eigen-decomposition $\Sigma = U\Lambda U^\top$ where U is a matrix of eigenvectors and Λ is a diagonal matrix with eigenvalues $\lambda_1, \ldots, \lambda_n$ along it. If the eigenvalues are sorted, the explained variance of the subspace formed by the first i eigenvectors is $1 - (\sum_{j=1}^{i} \lambda_j)/(\sum_{j=1}^{n} \lambda_j)$.

We also consider the Fourier basis which represents images using an orthonormal basis of complex sinusoids of different frequencies For natural images, pixel-pixel correlations are mostly only a function of the distance between them, i.e., Σ_{ij} only depends on $\left|i-j\right|$. Under this assumption, the eigenvectors of Σ are sinusoids (Hyvärinen et al., 2009). The Fourier Transform is, hence, closely related to PCA. We use the two-dimensional Discrete Fourier transform (Briggs et al., 1995; Oppenheim, 1999) to project each channel of the image. The Discrete Fourier transform (DFT) of an image x is

$$\hat{x}(\omega_1, \omega_2, c) = \frac{1}{d_1 d_2} \sum_{k=1}^{d_1} \sum_{l=1}^{d_2} x(k, l, c) \exp\left\{-2\pi i \left(\frac{\omega_1 k}{d_1} + \frac{\omega_2 l}{d_2}\right)\right\},$$

where ω_1, ω_2 are spatial frequencies. PCA and Fourier transform are a global statistic of the input image. Wavelets can provide local statistics at different scales using a set of basis functions that have compact support in both frequency and space (Mallat, 2008). We use the discrete 2D wavelet transform for all our experiments with Daubechies 4 wavelets from the PyWavelet package.

Slow feature analysis (SFA) for sounds. We used a cochlear model with 42 gammatone spectral filters followed by a temporal filter (Zhang et al., 2001; Lewicki, 2002; Tabibi et al., 2017) to approximate the structure of audio inputs in our task to that of the auditory system. Slow Feature Analysis ((Wiskott and Sejnowski, 2002)) argues that higher-order information in a stimulus (e.g., identity) changes at a slower time scale than other fluctuations (e.g., acoustic features). It has been shown to learn features similar to those in the V1 cortex (Berkes and Wiskott, 2005). If the auditory stimulus, e.g., output of the cochlear model, is $x(t)$, features found by (linear) SFA are eigenvectors of the covariance matrix of the derivative $\dot{x}(t)$, arranged from the smallest eigenvalue to the largest, i.e., slowest to fastest. SFA is equivalent to a PCA of the temporal derivative of a signal. section A.6.5.5 provides more details.

Shapley values (Shapley et al., 1953) calculate the improvement in the accuracy of a model with or without including a feature, averaged over all sets of other features (Lundberg and Lee, 2017). This requires fitting exponentially many models. But we can use an equivalent definition framed as a least squares problem with a linear constraint (Covert and Lee, 2020) and optimize it using dataset sampling (Ribeiro et al., 2016) to estimate the average improvement in accuracy by including a particular subspace, formed by different PCA

components, frequencies or scales. The Shapley values sum up to the accuracy of the model when all features are included.

Partial information decomposition (PID) (Williams and Beer, 2010) was introduced to understand how the mutual information $I(X_1, X_2; Y)$ of two random variables X_1, X_2 with Y can be decomposed into the redundant information R (both X_1 and X_2 have), synergistic information S (emerges only when both are available), and the unique information with respect to Y for both variables, denoted by U_1 and U_2 respectively. By definition $I(X_1, X_2; Y) = R + S + U_1 + U_2$.[10] We have

$$I(X_1; Y) = R + U_1, \text{ and } I(X_2; Y) = R + U_2. \tag{4.12}$$

PID is not uniquely defined using these constraints. But it turns out that defining redundant information is enough. Williams and Beer (2010) provide a formula for R but it is computationally intractable. However, if (X_1, X_2, Y) are jointly Gaussian, Barrett (2015) showed that the definition reduces to

$$R = \min \{I(X_1; Y), I(X_2; Y)\} ,$$

which can be computed using the formula for the mutual information of Gaussian random variables. Using redundancy R, we can calculate U_1, U_2 and S as well using eq. (4.12).

Estimating mutual information. Calculating the mutual information $I(X; Y) = H(Y) - H(Y \mid X)$ from samples requires a numerical estimate of the entropies. The Kraskov estimator (Kraskov et al., 2004) uses a k-nearest neighbor based estimate of the entropy and we additionally exploit the identity $I(X; Y) = H(X) + \sum_y P(Y = y) H(X \mid Y = y)$ for discrete Y. Upon this, one can implement a local non-uniformity correction (LNC) term (Gao et al., 2015), which is important when variables are strongly correlated. Belghazi et al. (2018) developed an estimator called MINE that uses a neural network to optimize the Donsker-Varadhan inequality. In general, estimating mutual information in high dimensions is difficult (Czyz et al., 2024). We only estimate it for labels and 10-dimensional PCA subspaces, so we expect our estimates to be reliable.

[10] Consider two bits $X_1 \in \{0, 1\}$ and $X_2 \in \{0, 1\}$. If $\{0, 1\} \ni Y = X_1$ xor X_2, then X_1 and X_2 contain no redundant or unique information about Y. But there is 1 bit of synergistic information. If $X_1 = X_2 = Y$, then X_1 and X_2 contain 1 bit of redundant information for Y.

4.2.2 Results

We next describe our findings using a broad range of evidence, analysis, and discussion. We use the CIFAR-10 (Krizhevsky, 2009) and ImageNet (Deng et al., 2009) for classification; the Cityscapes (Cordts et al., 2016) for semantic segmentation and depth estimation; the ADE20K for semantic segmentation (Zhou et al., 2017); and an augmented version of the M3ED dataset (Chaney et al., 2023) for optical flow, depth estimation and semantic segmentation. These tasks are diverse and complex, e.g., M3ED contains data from many natural scenes, including cars driving in urban, forest, and daytime/nighttime conditions.

We study these tasks using the PCA, Fourier and wavelet bases and seek to understand which basis elements are important for these tasks. We create different indices that order the elements of these bases. For PCA, we sort by explained variance, lower indices have high explained variance. For the Fourier basis, lower indices correspond to smaller radial frequencies. For the wavelet basis, lower indices correspond to the smallest scales. See Section A.6.5.2 for more details on how the indices are constructed.

4.2.2.1 Both input data and the task are effectively low-dimensional. Elements of the Fourier and wavelet bases have a large overlap with those of PCA.

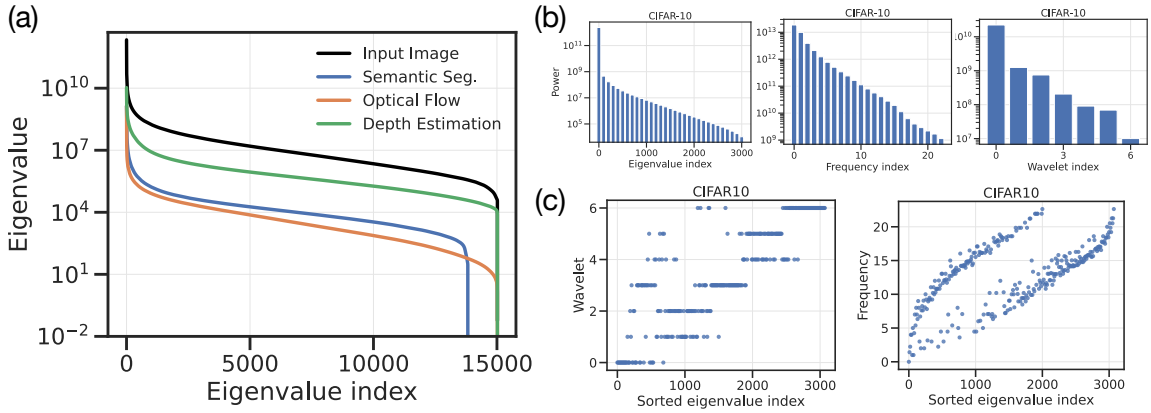


Figure 4.9: **(a)** Eigenvalues of the pixel-wise covariance matrix for inputs and outputs of different tasks are spread across a large range and decay quickly. **(b)** Variance or energy decays quickly with an increase in the index for PCA, Fourier and wavelet bases. **(c)** Index of wavelet or Fourier basis element (y-axis) that has the highest amplitude for images projected onto a PCA eigenvector of a particular index (x-axis). High Fourier and wavelet indices (large radial frequency and large scale, respectively) correspond to PCA eigenvectors with higher indices (or smaller eigenvalues).

Figure 4.9 (a) shows that for all datasets and modalities, inputs measured in different bases, namely pixel, Fourier, and wavelet space, exhibit a characteristic low-dimensionality. Eigenvalues are spread across a very large range—more than 10^6—for many of these tasks. When eigenvalues are sorted by their magnitude, there is a sharp "elbow", i.e., very few dimensions are necessary to capture most of the variance in the data, e.g., for CIFAR-10 which consists of 32×32 RGB images, the subspace corresponding to the first 5 eigenvalues of the pixel-wise covariance matrix has 90% explained variance and the remaining 3067 dimensions capture a mere 10% of the variance. The eigenspectra of all these datasets contain a long tail of eigenvalues that are spread linearly on a logarithmic scale. The numerical difficulty of any optimization algorithm, even linear regression, is governed by the condition number of the input covariance matrix (Boyd and Vandenberghe, 2004). The optimization problem underlying perception tasks is, therefore, ill-conditioned. This can result in long training times and under-fitting to the signal in the tail.

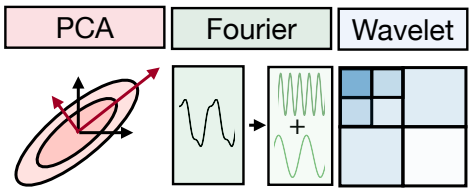


Figure 4.10: Schematic of Principal Components Analysis, Fourier and wavelet basis.

For some tasks, namely optical flow, depth estimation, and semantic segmentation, the ground-truth output can also be considered an image to calculate its eigenspectrum. Again, the characteristic low-dimensional pattern is evident in all bases. It is well known that amplitude spectra of natural images decay as $\sim 1/|f|$ with the frequency f. We see a similar phenomenon for dense output tasks. The new observation here is that the ground-truth labels of many tasks are also effectively low-dimensional.

PCA, Fourier, and wavelet are three different orthogonal bases. fig. 4.9 (b) shows that there is a large degree of overlap between inputs and outputs projected onto different subspaces in these bases, i.e., coefficients corresponding to the principal subspace of PCA are highly correlated with those of the Fourier basis at small radial frequencies or wavelets of small scales. Even if the three bases are linear transformations of the original pixel space, we would not have expected them to be aligned like this. Certainly, PCA is a dataset-dependent basis, while Fourier and wavelet bases are universal.

4.2.2.2 The principal subspace of the input data is most predictive of the task. However, the subspace with the least explained variance is also remarkably predictive. Even a random subspace is predictive of the task.

We trained deep networks on inputs projected onto different subspaces. We create these subspaces using "bands" of basis elements formed by the PCA, Fourier, and wavelet bases. Each band contains a contiguous block of basis elements sorted by their index. A low-band pass with an index i consists of a subspace that contains all eigenvectors/frequencies/scales with an index smaller than i. The principal subspace where the explained variance/energy per dimension is the highest (see section 4.2.2.1 (b)) lies in a low-band pass with a small index i. These bands are cumulative, so the principal subspace also lies in a low-band pass with a large index i. The high-band pass at index i is a subspace that consists of all eigenvectors/frequencies/scales with an index bigger than i. A high-band pass with a large index i contains the subspace with the smallest explained variance per dimension; a high-band pass with a small index i also contains this subspace. A band pass of index i corresponds to a subspace formed by a few sorted eigenvectors/frequencies/scales to the left and right of i with width specified for each experiment. An i-j band pass corresponds to a subspace formed by eigenvectors/frequencies/scales with an index between i and j. The subspace that corresponds to index i does not have any overlap with that of index $(i + 1)$[11].

Classification tasks. fig. 4.11 (a) and (c, green) show that different bands of the Fourier basis have remarkably high accuracy on ImageNet. In fig. 4.11 (a), the original image is unrecognizable to the human eye when projected into higher frequency bands. But a deep network can be trained on such images, and it gets 65% test accuracy in the 30–45 radial frequency band pass. fig. 4.11 (b, green) shows that the test accuracy on CIFAR-10 is largely the same for different bands for the Fourier (middle) and wavelet (right) bases, but it drops for PCA (left).[12] In short, the predictive ability of a band is correlated with its explained variance/power. However, many of these bands are non-trivially predictive of the task. For CIFAR-10, the last band pass of PCA has 20 % accuracy. fig. 4.11 (b, orange) shows that the test accuracy increases with the low pass index

[11] A Butterworth (Butterworth, 1930) filter of order 5 is used to avoid ringing effects produced from a box filter; the cutoff frequency is set at the 3 dB point. Due to this, there is a small overlap in our bands for the Fourier basis; there is no such overlap in the PCA or wavelet bands.

[12] This can be understood by looking at fig. A.17. As the index increases, the number of orthogonal components in the PCA band pass is unchanged, so the explained variance of different bands decreases. But for Fourier and wavelet bases, the number of frequencies/scales increases sharply with the index. This is because high radial frequencies span a large number of spatial frequencies.

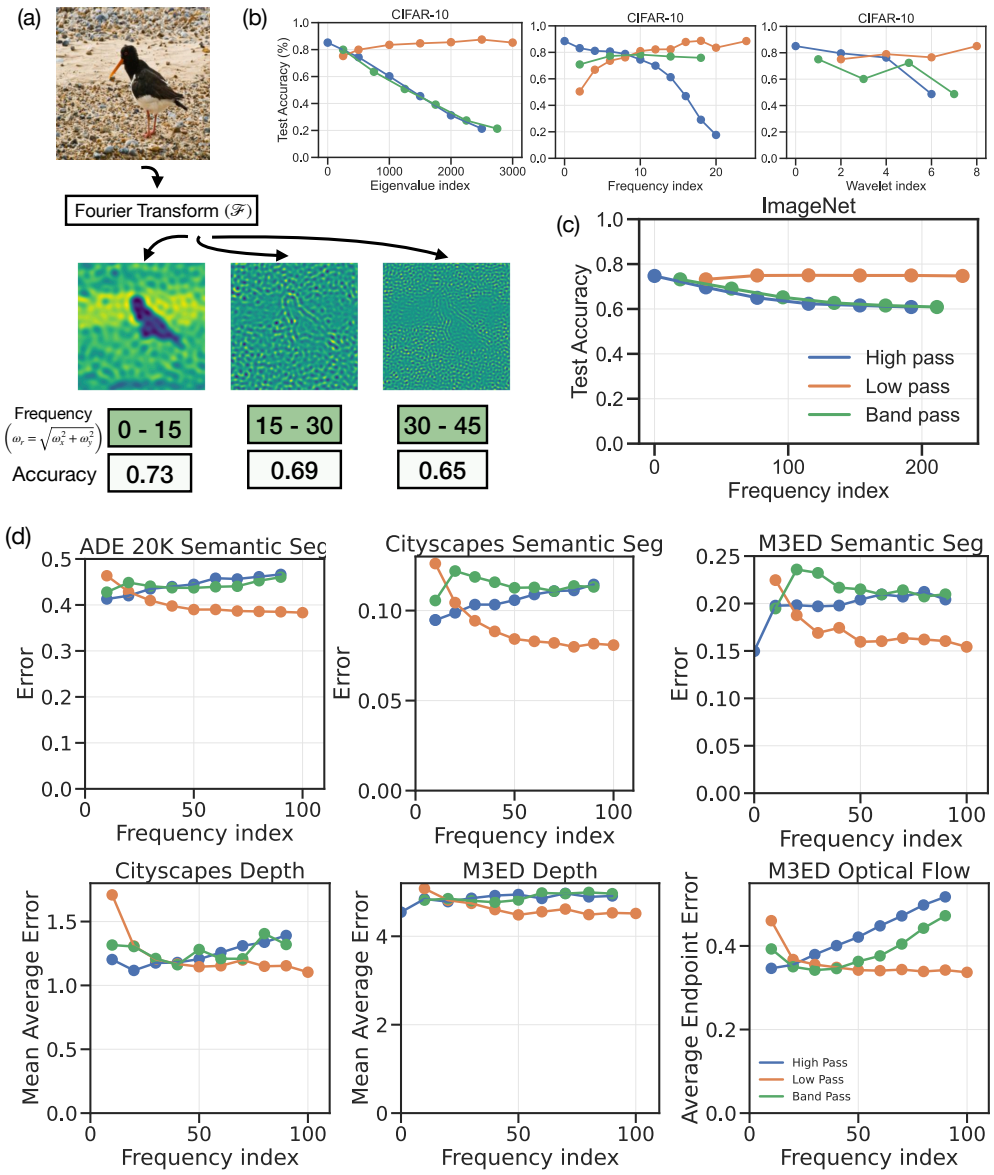


Figure 4.11: Panel **(a)** shows that the image, when projected on a high frequency band (30–45) cannot be recognized by the human eye; and yet a network trained on such images can get more than 65% test accuracy. We show the test accuracy (for CIFAR10 **(b)** and ImageNet **(c)**) of networks trained on images projected onto different subspaces. Remarkably, for ImageNet, all frequency bands achieve more than 60% accuracy. Almost all PCA subspaces, radial frequencies and scales are useful for image classification on CIFAR-10 and ImageNet; observe that low pass, band pass and low-index high pass regimes all obtain good test accuracy. However, the head of the spectrum usually contains more discriminative information than the tail. **(d)** For dense perception tasks such as semantic segmentation, optical flow and depth prediction, the results are consistent with classification, i.e., the information for the task is also present redundantly across the spectrum. Many frequency bands result in remarkably low errors on these tasks. Error barely improves with index for low pass filters, indicating diminishing returns on these tasks as higher frequencies are included in the data.

i as the dimensionality of the subspace grows; this trend also holds for ImageNet in fig. 4.11 (c, orange). Adding new basis elements, eigenvectors, frequencies, or scales has diminishing returns on accuracy. This

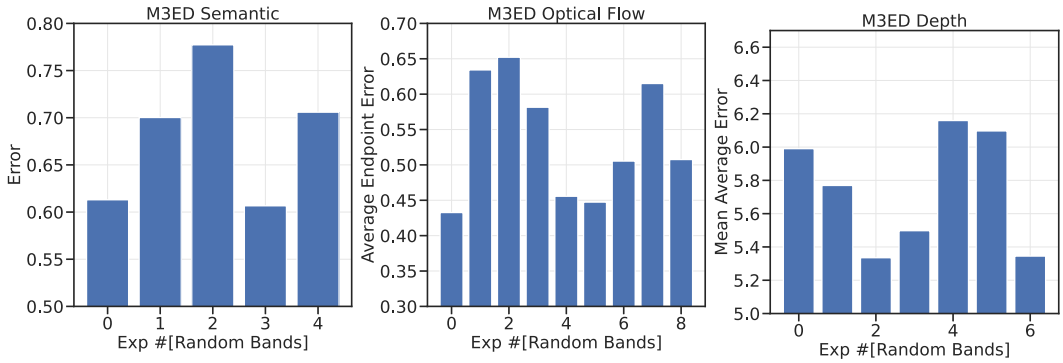


Figure 4.12: Error on perception tasks is remarkably low even when networks are trained on images projected onto a random linear subspace; see A.6.5.4 for the parameters of the randomly chosen center frequencies and widths for the band pass filter. This suggests that information for the task is present throughout the spectrum. The "explained" power is less than 20% for all bands of randomly chosen frequencies, for all three panels.

trend is exactly reversed for high pass—accuracy drops with the index i as the explained variance/power of these subspaces decreases. fig. 4.12 shows that band passes created from a random subset of the basis are also predictive of the task.

Therefore, the principal subspace is usually the most predictive (green curves decrease with index, except for the Fourier basis). The tail of the spectrum also has remarkable predictive ability. Essentially, any subspace of the input data is predictive of these tasks.

The above trends also hold for optical flow estimation, depth estimation and semantic segmentation. See fig. 4.11 (d).[13] For optical flow estimation, the Average Endpoint Error (AEE) is smaller for low frequency bands (orange) than high frequency bands (blue). Error decreases and then saturates with increasing index for low pass filters (orange). AEE for high pass filters (blue) continues to increase, essentially linearly, with the index. This might indicate that the low-frequency spectrum contains more predictive features than the high frequency spectrum, where, we suspect, features are redundant for optical flow. This is consistent with existing results that have argued that low-frequency information is important for motion perception (Shi et al., 2020). Trends for semantic segmentation and depth estimation are similar. Similar to flow, previous literature on depth has shown the importance of low-frequency content for depth estimation (Kane et al., 2014). Error (1-Accuracy) quickly saturates for low pass filters (orange) and decreases roughly linearly for high pass filters (green). Error of different bands (blue) increases slightly with frequency.

[13] Also see fig. A.19 for experiments with a different neural architecture and different filters for some of the tasks.

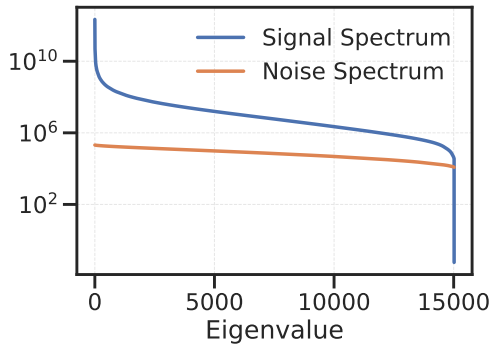


Figure 4.13: The amplitude of noise in images from the M3ED dataset is roughly constant across the spectrum and smaller than the signal, even in the tail. Signal-to-noise ratio (SNR) is larger in the head than in the tail.

The signal is stronger than the noise, even in the tail. Statistical wisdom is to avoid using the tail of the spectrum for making predictions due to noise (Pyatykh et al., 2013; Chen et al., 2015; Gavish and Donoho, 2014) either due to low signal coming from unobserved data (rare categories), or due to high noise of the acquisition process (camera). Surprisingly, the tail of the spectrum is predictive for so many tasks. We used the method of Rakhshanfar and Amer (2016), which estimates the level of additive white Gaussian noise in the image by finding homogeneous patches in the image frame to estimate the variance of intensity there. For one of our datasets, M3ED, the variance is $\sigma^2 = 1.41$. This is shown in fig. 4.13 along with the original spectrum for comparison. As expected, the signal-to-noise ratio (SNR) is larger in the head. But the noise magnitude is constant over different eigenvalues, and it is well below the magnitude of the smallest eigenvalue.

We make a similar observation from the experiments for fig. 4.14. We use bootstrap sampling to create multiple datasets with $n = 5000, 10000, 20000$ samples drawn from CIFAR10. We create 50 datasets for each dataset of size n. For each dataset we compute its eigenvectors, which are divided into 10 bands. Each band of d eigenvectors defines a d-dimensional subspace and the similarity between two bands of eigenvectors V_1 and V_2 is $\frac{1}{d}\| V_1^T V_2\|_F$; the similarity lies between 0 and 1 depending on the extent of overlap between the two subspaces. In fig. 4.14, we compute the average similarity between every band of eigenvectors computed using one dataset to that computed using another dataset. The bands of eigenvectors span similar subspaces even if they are computed using different datasets.

Redundancy in vocalization discrimination tasks. Bregman (1990) showed that temporal regularity, i.e., patterns in frequency changes over time, plays a critical role in auditory perception. DiTullio et al. (2023)

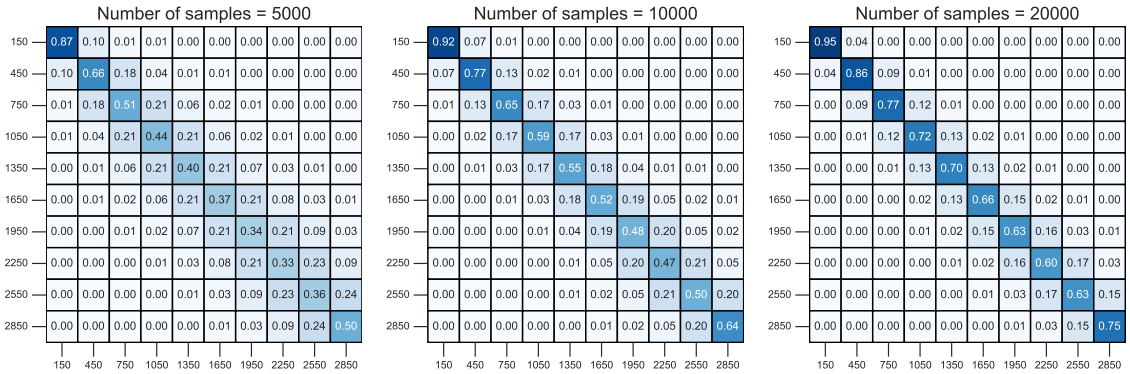


Figure 4.14: We plot the average similarity between bands of eigenvectors estimated from two different datasets. Both the head and the tail are similar across different draws of the dataset, as indicated by the diagonal entries of the matrix. The head of the spectrum is more stable than the tail. This plot suggests that the entire spectrum is stable and that the signal is stronger than the noise even in the tail.

argued that this could be because of temporal regularities in vocalization; slowest frequencies can be used to discriminate between rhesus macaque vocalizations.

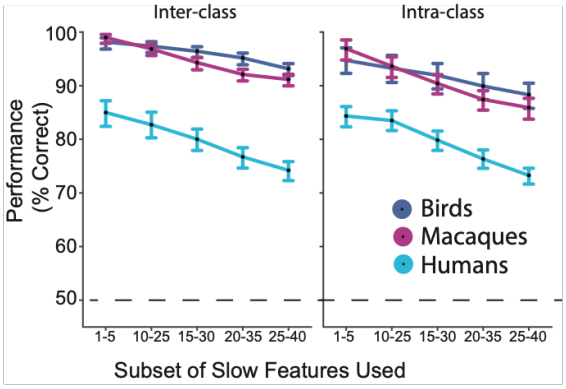


Figure 4.15: Vocalizations can be discriminated well, using even fast SFA features. Accuracy across 200 pairs of vocalizations (error bars are 95% confidence intervals). Left figure is for inter-class pairs, i.e., vocalizations from two different human speakers, macaque speakers, or species of bird. Right figure is for intra-class pairs, i.e., different numerical digits from the same human speaker, utterances from the same macaque, or songs from the same species of bird.

We are interested in understanding if faster temporal frequencies can also be used for this task. We studied bird songs of four species (Zhao et al., 2017), macaque vocalizations (Fukushima et al., 2015), and human vocalizations of MNIST digits (Becker et al., 2024) using slow feature analysis (SFA) (Wiskott and Sejnowski, 2002). SFA features correspond to the eigenvectors corresponding to the smallest eigenvectors in the time derivative of the auditory stimulus. fig. 4.15 shows that a multi-layer perceptron trained using faster features can also discriminate between different vocalizations quite well. Performance decreases quite slowly as faster features are used (the confidence intervals overlap), and it is well above chance. This observation is consistent

with visual perception tasks.

4.2.2.3 Different PCA subspaces have redundant and synergistic information about the classification task with very little unique information.

We next use the classification task to see that the observation above, although surprising, may not be due to any specific properties of deep networks. It seems to be inherent to the data.

Labels have high mutual information with every subspace of the input data. fig. 4.17 (a) shows the mutual information $I(Y; P_i X)$ of labels Y with input images X from CIFAR-10, that are projected onto the i^{th} eigenvector of the pixel-wise input covariance matrix; section 4.2.1 gives more details. The Kraskov estimator (Kraskov et al., 2004) calculates the mutual information of the discrete labels with the (scalar) projected images, using a k-nearest neighbor estimator of the entropy; LNC stands for local non-uniformity correction of the entropy estimator (Gao et al., 2015). MINE (Belghazi et al., 2018) is a neural-network based estimator of mutual information. Both these estimates of $I(Y; P_i X)$ suggest that the principal subspace has high mutual information with the labels.

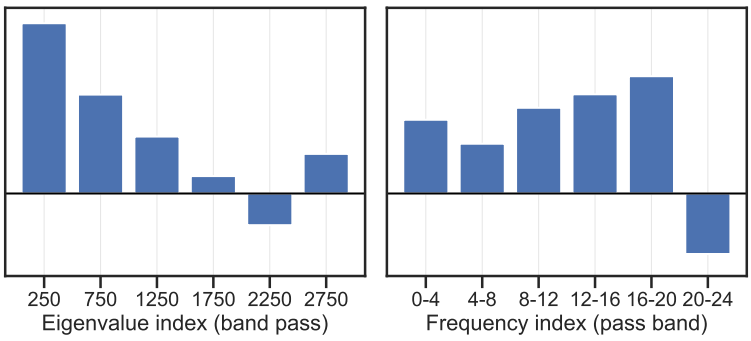


Figure 4.16: SHAP values of different PCA and frequency bands: The head of the spectrum is the most important, but the tail is also necessary for classification on CIFAR-10. The horizontal black line corresponds to a SHAP value of 0.

The mutual information is also large for eigenvectors in all the other subspaces. The explained variance of the high-index eigenvectors is very small, so one might expect $I(Y; P_i X)$ to decay strongly for those. In short, any subspace of the input data should be predictive of the ground-truth labels, fig. 4.11 provided experimental evidence for this. This finding is corroborated by SHapley Additive exPlanations (SHAP) values in fig. 4.16; also see section 4.2.1.

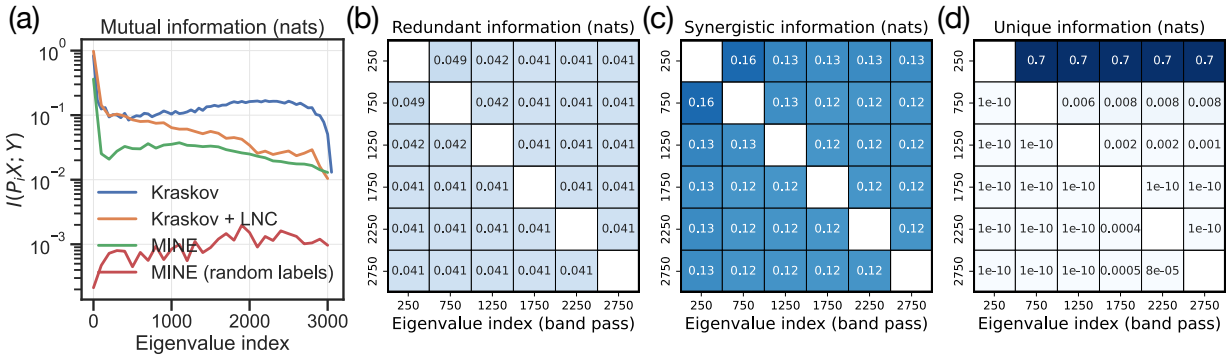


Figure 4.17: **(a)** Images projected onto different PCA subspaces (each consisting of 10 contiguous basis elements) have non-trivial mutual information with the labels, across the spectrum. Kraskov and Kraskov-LNC are non-parametric estimators of mutual information, MINE uses a neural network(Belghazi et al., 2018). While the principal subspace has the highest mutual information with the labels, the tail also has non-trivial mutual information. For comparison, the mutual information with randomly permuted labels is much smaller. Partial information decomposition for CIFAR-10 suggests that different bands have high amounts of redundant **(b)** and synergistic information **(c)**. Cell (i, j) corresponds to the PID decomposition for the task's subspaces i and j. Unique information is much smaller **(d)** for anything other than the principal subspace. We note that (b,c,d) assume that inputs and labels are jointly Gaussian, which could be a poor approximation. Fourier and wavelet basis trends are similar; see fig. A.13.

Partial information decomposition for identifying the redundant, unique and synergistic information. Both the eigenspectrum's head and tail have information pertinent to classification, but we do not know if this is the same kind of information. We used the concept of partial information decomposition (PID (Williams and Beer, 2010)) to investigate this. Given variables X_1 and X_2, PID decomposes the mutual information $I(X_1, X_2, ; Y)$ into redundant (R), synergistic (S) and unique information (U_1 and U_2 corresponding to X_1 and X_2 respectively). The total mutual information $I(X_1, X_2; Y) = R + S + U_1 + U_2$.[14,15] Different PCA bands correspond to random variables X_1 and X_2 for us. The mutual information of each band decomposes as $I(X_1; Y) = R + U_1$. The fact that $I(X_1, Y)$ for different eigenvectors in fig. 4.17 (a) has a comparable magnitude as that of any cell in fig. 4.17 (b) suggests that different PCA bands of CIFAR-10 have a lot of redundant information with the labels. This is borne out by fig. 4.17 (d), which shows that low-index PCA bands (high explained variance) have larger unique information than bands in the tail. Synergistic information is usually harder to interpret. In this case, it is large for any two PCA bands. This analysis suggests that the observations in fig. 4.11 are due to inherent properties of the input data and the ground-truth labels.

[14] We discuss PID in section 4.2.1 further. Redundancy is the minimum information about Y provided by either variable; it is independent of correlation between the variables. Synergy S is the extra information contributed by the weaker source when, the stronger source is known and can either increase or decrease with correlation between sources; typically jointly Gaussian random variables are net synergistic (Barrett, 2015).

[15] It is difficult to calculate PID, or its variants, using samples (Latham and Nirenberg, 2005). We approximated that X_1, X_2, and Y are jointly Gaussian. Note that the ground-truth label Y is a categorical random-variable. But we nevertheless plough forward with this approximation. It is reassuring that this analysis corroborates fig. 4.11, so this approximation is not entirely invalid.

4.2.2.4 Despite redundant and synergistic information across the entire spectrum, a deep network predominantly uses information in the head

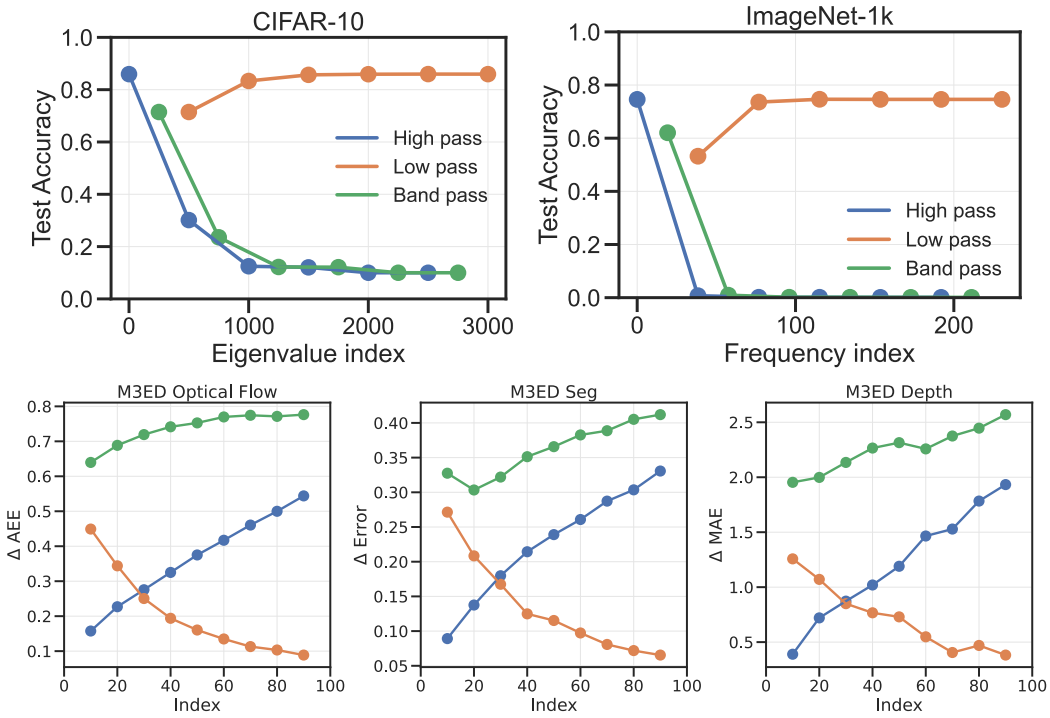


Figure 4.18: Trained networks predominantly use information present in the head of the spectrum. They largely ignore the tail. **Top**: Accuracy on ImageNet drops to 0.1% if we exclude radial frequencies smaller than 30 at test time. **Bottom**: For regression tasks, we show the difference between the error of a network trained on every band with a network trained on a particular band; the error for both networks is computed using test images projected onto a particular band. Error differential increases with index for band pass and high pass; the network, therefore, predominantly uses information in the head. Also, see fig. A.15 for a similar result using features from different layers in a trained network.

The performance of low pass bands improves with the index in fig. 4.11, so there is some synergistic information in the different subspaces (fig. 4.17 (c)). Therefore, the ideal learner would build features from all subspaces, discarding redundant information and selecting the synergistic and unique parts. It is natural to ask whether a deep network behaves like this. We trained networks using the original images but tested them on images projected into different subspaces, e.g., low pass means that test images were projected into the low pass subspace of that index, and similarly for band pass and high pass. Bands computed from PCA were used for CIFAR-10 classification, and the Fourier basis was used for ImageNet and other tasks.

Classification accuracy drops to chance for both CIFAR-10 and ImageNet if the network does not have access to lower bands; see fig. 4.18 (top). For regression tasks, we plotted things differently and compared the error

of a network trained on all bands but tested on a particular band to the error of a network trained on a particular band from fig. 4.11 (d). If the error differential is large for a band, then the network trained on all the bands does not use information in that particular band. fig. 4.18 (bottom) shows that the error differential increases for band pass and high pass curves, i.e., networks predominantly use information in the head of the spectrum. Unsurprisingly, the condition number of the optimization problem underlying these tasks is extremely large (fig. 4.9). A large number of weight updates are necessary to fit the small amount of signal in higher bands.

4.3 Related Work

Understanding the space of learnable tasks A large body of work has sought to characterize relationships between tasks, e.g., domain specific methods (Zamir et al., 2018; Cui et al., 2018; Pennington et al., 2014), learning theoretic work (Baxter, 2000; Maurer, 2006; Ben-David et al., 2010; Tripuraneni et al., 2020a; Hanneke and Kpotufe, 2020; Caruana, 1997), random matrix models (Wei et al., 2022a), neural tangent kernel models (Malladi et al., 2022) and information-theoretic analyses (Jaakkola and Haussler, 1999; Achille et al., 2019a,b). Broadly speaking, this work has focused on understanding the accuracy of a model on a new task when it is trained upon a related task, e.g., relationships between tasks are characterized using the excess risk of a hypothesis. Our methods also allow us to say things like "task P^1 is far from P^2 as compared to P^3". But they can go further. We can glean a global picture of the geometric structure in the space of tasks and quantify statements such as "the divergence between P^1 and P^2 eventually is more than that of P^1 and P^3, but representations learned on these tasks are similar for 30% of the way".

There is strong structure in typical inputs, e.g., recent work on understanding generalization (Yang et al., 2022; Bartlett et al., 2020) as well as older work such as Simoncelli and Olshausen (2001); Field (1994); Marr (2010) has argued that visual data is effectively low-dimensional. Our works suggests that tasks also share a low-dimensional structure. Just like the effective low-dimensionality of inputs enables generalization on one task, effective low-dimensionality of the manifold of models trained on different tasks could perhaps explain generalization to new tasks.

Natural image statistics. Natural data is statistically redundant (Simoncelli and Olshausen, 2001). The amplitude spectrum for both luminance (Atick and Redlich, 1992; Van der Schaaf and van van Hateren, 1996; Field, 1987) and color (Burton and Moorhead, 1987) falls off as $\sim 1/\left|f\right|$ with frequency f in natural images; this pattern is consistent across scales. Spatio-temporal statistics follow similar trends (van Hateren, 1992; Dong and Atick, 1995; Olshausen, 2000). Edgust the inputs. Typical tasks are also redundant functions of the input. The former is due to regularities in natural environments. While the latter is, perhaps, a property of tasks that biological organisms and machines *chose to do*. Tasks that are ecologically and economically useful.

Removing redundancy in the stimuli is fundamental to how the brain works (Barlow et al., 1961; Attneave, 1954). There are circuits that implement whitening, low-pass filtering and compression (Laughlin, 1981; Atick and Redlich, 1992; Barlow, 1989), sparse coding (Barlow, 1972; Olshausen and Field, 1997; Field, 1993; Chechik et al., 2001), slow feature analysis (Berkes and Wiskott, 2005; Wiskott and Sejnowski, 2002) etc. But the efficient coding hypothesis could not be the complete story because there are three things at play: the environment, metabolic and architectural constraints imposed by the neural circuitry, and the task (Simoncelli and Olshausen, 2001; Field, 1994; Balasubramanian and Sterling, 2009). Although we know some principles to explain how neural circuits adapt to the environment (Ratliff et al., 2010; Balasubramanian, 2015), in spite of intense activity there is no clear understanding of task informs learned representations (Ramesh and Chaudhari, 2022; Mao et al., 2024). Normative principles of representation learning (Tishby et al., 1999; Achille and Soatto, 2018; Barlow, 2001; Yerxa et al., 2023; Yu et al., 2020) are, as yet, inadequate.

We showed that evidence for the task is spread across different parts of the observation space. If this evidence were not redundant, tasks would be difficult to learn—both due to the condition number of the input spectrum and because of noise in the tail. But because the evidence of the task is redundant, an organism, or a machine, need not be very careful as to which subspace it uses. Learning, ontogenetic or over evolution, will refine neural circuitry and the representation, to improve performance. There is also a second reason why this redundancy is useful. Without it, a specific set of features would have to be learned for each task. Changes in the task would severely impair the organism until the requisite features were learned.

Implications for theoretical questions in deep learning. Consider logistic regression to fit labels $y_i = \operatorname{sign}\langle w^*, x_i\rangle$ using a linear model $\hat{y}_i = \operatorname{sign}\langle w, x_i\rangle$. Even if the problem is strongly convex, gradient descent requires $O(\kappa \log(1/\epsilon))$ iterations to reach ϵ-optimality where κ is the condition number of $\sum_i x_i x_i^\top / n$ (Bottou et al., 2016). For all tasks in this paper, $\kappa > 10^6$. Therefore, gradient descent evolves predominantly in the principal subspace of the input covariance matrix. Spectral bias has been studied for linear models (Hacohen and Weinshall, 2022), kernel machines (Yao et al., 2007), neural networks (Rahaman et al., 2019; Tancik et al., 2020) or the neural tangent kernel (Cao et al., 2021). In short, effectively low-dimensional inputs result in ill-conditioned optimization problems (Ma and Belkin, 2017).

Whitening risks amplifying noise, or spurious correlations from finite samples.

These works typically assume that the task needs all the features, or that it needs a few specific features. But this does not explain why optimization problems underlying perception tasks can be solved so effectively in practice in spite of the large condition number. If the task is redundant, say, $y_i = \text{sign}\langle w^*, Px_i\rangle$ for many matrices P which project data into different subspaces, then since $\langle w^*, Px_i\rangle = \langle Pw^*, x_i\rangle$, there are many equivalent solutions and weight initializations with a large overlap with one of them. Our results indicate that networks pick a particular projection P that emphasizes the head of the input spectrum.

Alignment between the labels and the principal subspace of the inputs improves generalization for both kernel methods (Amini et al., 2022) and two-layer networks (Arora et al., 2019). If input data are effectively low-dimensional, then one can obtain analytical generalization bounds for deep networks (Yang et al., 2022; Bartlett et al., 2020), as also generalize well in practice (Pope et al., 2021; Martin and Mahoney, 2021). This also leads to some dramatic phenomena: networks of different architectures, training and regularization methods, evolve on extremely low-dimensional manifolds (Mao et al., 2024). Broadly, learning theoretic investigations do not consider the redundancy in the task. Our observations suggest that this may be an important direction to investigate.

Implications for practice. Reconstruction-based methods, e.g., masked auto-encoders (MAE), often take longer to train, and fall short of the accuracy obtained by contrastive learning-based methods on downstream tasks. Balestriero and LeCun (2024) explained this by (a) arguing that the tail of the spectrum is important for classification, (b) training MAEs is slow due to the condition number of the reconstruction problem, and (c) MAEs could not learn useful features for perception. Our numerical results are not at odds with theirs. It is simply that for the explained variance of the tail to be as high as that of the head, one needs to use a very large number of eigenvectors—essentially all of them. Our results paint a very different story. The principal subspace is most important for the task. But even a random subspace can predict the task remarkably well. Therefore, while training MAEs is slow, perhaps, due to the input data being low-dimensional, the performance gap compared to contrastive learning may be for other reasons. Learning by reconstruction does produce features that are informative for perception—masking performs decorrelation in the Fourier basis.

CHAPTER 5

CONCLUSION

This dissertation studies the principles of learning from multiple tasks using perspectives from learning theory, information theory and signal processing. We make progress towards answering two questions: (i) How do we optimally train representations using data from multiple tasks? (ii) What characterizes "typical" learnable tasks?

We use our theory of task competition and reference priors to develop Model Zoos for labeled and unlabeled data — we train many small models that span the space of tasks as opposed to one model on all the data. Our small-scale experiments suggest that most existing networks are not trained optimally, despite generalizing incredibly well to many tasks. Our theory suggests that we can generalize better if we carefully curate which data to train on or weigh the losses on individual samples.

In the second half of the dissertation, we attempt to characterize typical learnable tasks. We find that many perception tasks are highly redundant functions of the input, i.e. many different and even disjoint subspaces can be used to predict the label with remarkably high accuracy. We speculate that this redundancy enables learning, i.e. any random subspace allows you to solve the task and signal for the task is not sparse. We believe that a sensor that captures highly redundant inputs allows agents to tackle many tasks and build a shared representation for them. We chose to tackle redundant tasks because those are the only tasks that agents with bounded resources can readily solve and tasks that machines are made to solve.

APPENDIX A

ADDITIONAL RESULTS

A.1 Theoretical results for Prospective learning

A.1.1 Bayes risk for a Markov chain

We would like to compute the prospective Bayes risk, when the evolution of the samples is governed by a Markov transition matrix where $P(Y_{t+1} = 0 \,|\, Y_t = 0) = \theta_0$ and $P(Y_{t+1} = 1 \,|\, Y_t = 1) = \theta_1$, i.e., the transition matrix is

$$\Gamma = \begin{bmatrix} \theta_0 & 1-\theta_0 \\ 1-\theta_1 & \theta_1 \end{bmatrix}.$$

The probability distribution at time t' is given by $\Gamma^{t'-t}(z_t, 1-z_t)^T$. The eigenvalues of the transition matrix

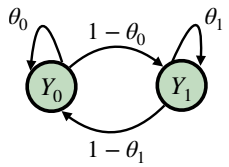


Figure A.1: Markov chain describing the evolution of data

are $\lambda_1 = 1$ and $\lambda_2 = \theta_0 + \theta_1 - 1$ with the corresponding eigenvectors being $(1,1)^\top$ and $(\theta_0 - 1, 1-\theta_1)^\top$. Diagonalizing Γ we get

$$\begin{aligned}\Gamma^{t'-t} &= \begin{bmatrix} 1 & \theta_0 - 1 \\ 1 & 1-\theta_1 \end{bmatrix} \begin{bmatrix} \lambda_1^{t'-t} & 0 \\ 0 & \lambda_2^{t'-t} \end{bmatrix} \begin{bmatrix} 1 & \theta_0 - 1 \\ 1 & 1-\theta_1 \end{bmatrix}^{-1} \\ &= \frac{1}{(2-\theta_0-\theta_1)} \begin{bmatrix} 1-\theta_1+(1-\theta_0)\lambda_2^{t'-t} & (1-\theta_0)-\lambda_2^n \\ 1+\theta_1+(1-\theta_0)\lambda_2^{t'-t} & (1-\theta_0)+\lambda_2^n. \end{bmatrix}\end{aligned}$$

which implies that the probability distribution of the state at time t' is

$$\pi_{t'} = \frac{1}{(2-\theta_0-\theta_1)} \begin{bmatrix} 1-\theta_1 \\ 1-\theta_0 \end{bmatrix} + \frac{\lambda_2^{t'-t}\left((1-\theta_0)(1-z_t)-(1-\theta_1)z_t\right)}{(2-\theta_0-\theta_1)} \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

Hence, the optimal sequence of hypotheses is $h^*_{\geq t+1} = (h^*_{t+1}, h^*_{t+2}, \dots)$, where

$$h^*_{t'} = \operatorname*{argmax}_{i \in \{0,1\}} \pi_{t'}(i)$$

with Bayes risk equal to $R^*_t = \lim_{T\to\infty} \frac{1}{T} \sum_{s=t}^{T} \min_{i\in\{0,1\}} \pi'_t(i)$. This reduces to

$$R^*_t = \frac{1}{(2-\theta_0-\theta_1)} \min(1-\theta_0, 1-\theta_1);$$

the second term in the expression of $\pi_{t'}$ vanishes as $T \to \infty$. If $\theta_0 = 0.9$ and $\theta_1 = 0.5$, then $R^*_t = 1/6$.

If we restrict our attention to the case where $\theta_0 = \theta_1$, the discounted Bayes risk reduces to

$$(1-\gamma)\sum_{s=t+1}^{\infty} \gamma^{s-t-1}\ell(h^*_s) = (1-\gamma)\sum_{s=t+1}^{\infty}\left(\frac{\gamma^{s-t-1}}{2} - \frac{\gamma^{s-t-1}\left|\lambda_2^{s-t}\right|}{2}\right) = (1-\gamma)\left(\frac{1}{2(1-\gamma)} - \frac{|\lambda_2|}{2(1-|\lambda_2|\gamma)}\right)$$

Substituting $\theta_0 = \theta_1 = 0.1$, the discount risk for $\gamma = 0.9$ is 0.357.

A.1.2 Proof of proposition 1

Let $\mathcal{X} = \{-1, 1\}$ and $\mathcal{Y} = \{0, 1\}$. Consider two distributions P_1 and P_2 (fig. A.2):

$$P_1(X = x) = P_2(X = x) = \frac{1}{2} \quad \forall x,$$

$$P_1(Y = 1 \mid X = x) = \begin{cases} \theta & \text{if } x = 1 \\ 1-\theta & \text{if } x = -1, \end{cases}$$

$$P_2(Y = 1 \mid X = x) = \begin{cases} 1-\theta & \text{if } x = 1 \\ \theta & \text{if } x = -1, \end{cases}$$

In other words, the inputs have the same marginals but the labels are flipped between P_1 and P_2. Consider a stochastic process Z such that $Z_{2t+1} \sim P_1$ and $Z_{2t} \sim P_2$ where $t \in \mathbb{N}$.

Let $\mathcal{G}$ be any hypothesis class and let $\ell(s, \hat{y}, y) = \mathbf{1}(\hat{y} \neq y)$ be the time-invariant zero-one loss. The time-agnostic learner uses a sequence of hypotheses $h \equiv (h_t)$ where $h_t = h_{t'}, \forall\, t, t' \in \mathbb{N}$ to make predictions at all times. The future loss is

$$\bar{\ell}_t(h, Z) = \lim_{\tau \to \infty} \frac{1}{2\tau} \sum_{s=t+1}^{t+2\tau} \ell(s, h_s(X_s), Y_s) = R_1(h) + R_2(h) = \frac{1}{2},$$

almost surely; here $R_1(h)$ and $R_2(h)$ are risks on data from distributions P_1 and P_2 at odd and even times, respectively. The last equation follows from the fact that $R_1(h) = 1 - R_2(h)$ because the labels are flipped. Prospective Bayes risk is zero if the hypothesis class $\mathcal{G}$ contains the Bayes optimal hypotheses for each of the two distributions. The future loss evaluates to $1/2$ for all realizations and so does the prospective risk. The prospective risk of a hypothesis sequence that makes random predictions (zero or one with equal probability at each instant) is also $1/2$. This stochastic process is not weakly prospective learnable.

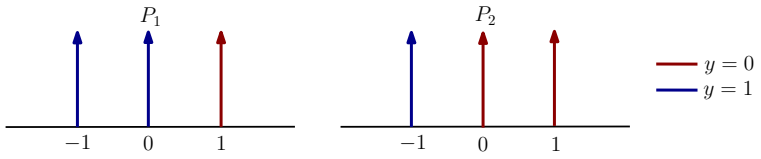


Figure A.3: A simple stochastic process that is weakly but not strongly prospectively learnable.

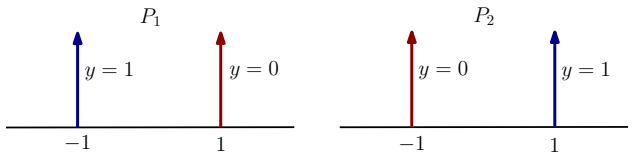


Figure A.2: A simple stochastic process that is not weakly prospectively learnable.

Now consider the two distributions shown in fig. A.3,

$$P_1(X = x) = P_2(X = x) = \frac{1}{3} \quad \forall x,$$

$$P_1(Y = 1 \mid X = x) = \begin{cases} \theta & \text{if } x \leq 0 \\ 1 - \theta & \text{if } x = 1, \end{cases}$$

$$P_2(Y = 1 \mid X = x) = \begin{cases} 1 - \theta & \text{if } x \geq 0 \\ \theta & \text{if } x = -1. \end{cases}$$

Inputs are supported on the set $\{-1, 0, 1\}$ this time. Again consider a stochastic process Z such that $Z_{2t+1} \sim P_1$ and $Z_{2t} \sim P_2$ for $t \in \mathbb{N}$. For a time-agnostic learner, since its hypothesis h at each time step has to predict incorrectly at $x = 0$, we have $R_1(h) + R_2(h) \geq \frac{1}{3}$. The future loss is

$$\bar{\ell}_t(h, Z) = \lim_{\tau \to \infty} \frac{1}{2\tau} \sum_{s=t+1}^{t+2\tau} \ell(s, h(X_s), Y_s) = R_1(h) + R_2(h) \geq \frac{1}{3}.$$

almost surely. It follows that the prospective risk $R_t(h) \geq \frac{1}{3}$ for any hypothesis. Prospective Bayes risk is again zero and therefore this stochastic process is not strongly prospectively learnable. It is however weakly learnable.

A hypothesis that predicts $\hat{y} = \pm 1$ with equal probability has $R_t^0 = 0.5$. If the data contains samples for $x \in \{-1, 1\}$, ERM will select a hypothesis that minimizes the empirical risk which necessitates that $h(1) = 0$ and $h(-1) = 1$. Therefore $R_1(h) + R_2(h) \leq \frac{1}{3} + \epsilon$, since h predicts correctly at $x = \pm 1$, and incorrectly at $x = 0$ exactly one of the two distributions. The constant ϵ can be chosen to be $\propto t^{-1/2}$ after receiving data from t timesteps. The probability with which we do not get samples at $x = 1$ or at $x = -1$, is 2×3^{-t}. Therefore the probability that $R_1(h) + R_2(h) \leq \frac{1}{3} + \epsilon$ is at least $1 - 3^{-t+1}$ after t time steps. This learner is therefore better than the chance learner whose risk is R_t^0 and it is a weak prospective learner. This shows that there exist stochastic processes that are weakly prospective learnable using time-agnostic ERM but not strongly.

A.2 Theoretical results for linear masked autoencoders

A.2.1 Linear MAEs: Characterizing critical points

We present some more details for theorem 21 below.

Consider the loss.

$$\ell = \| X - (1-m)XAB\|^2 + m(1-m)\| GAB\|^2, \tag{A.1}$$

We first set $\partial\ell/\partial A$ to 0 to get

$$-2(1-m)X^\top XB^\top + 2(1-m)^2 X^\top XABB^\top + 2m(1-m)\text{Blkdiag}_p(X^\top X)ABB^\top = 0. \tag{A.2}$$

Any critical point must satisfy

$$A^* = V^{-1}X^\top XB^\top(BB^\top)^{-1} \tag{A.3}$$

where $V = (1-m)X^\top X + m\text{Blkdiag}_p(X^\top X)$. Substituting this value of A^* back into the loss, we get

$$\ell = \text{Tr}\big(X^\top X\big) - (1-m)\,\text{Tr}\big[B(X^\top XV^{-1}X^\top X)B^\top(BB^\top)^{-1}\big]$$

Let $C = X^\top XV^{-1}X^\top X$ and $D = I$. Note that C and D are both symmetric and D is invertible. Using lemma 3, the expression is minimized by the k largest eigenvalues of the generalized eigenvalue problem defined on (C, D). Furthermore, every critical point is a subset of k eigenvectors (from Lemma 2).

A.2.2 Non-linear MAEs using linear approximations

Non-linear masked autoencoders under a Taylor series approximation. Consider a nonlinear autoencoder f and the corresponding masked autoencder loss

$$\ell_m = \mathbb{E}_R\| X - f(R \odot X)\|^2.$$

Let $X_\mu = (1-m)X$ and $X_r = R \odot X$. Under the Taylor series approximation around 0

$$f(X_R) \approx f(0) + X_R \nabla f(0)^\top = X_R \nabla f(0)^\top,$$

the MAE loss reduces to

$$\ell_m = \| X - (1-m)X\nabla f(0)^\top \|^2 + m(1-m) \| G \nabla f(0)^\top \|^2$$

Note that this approximation holds for small perturbations to the input, and is less likely to hold for large perturbations, i.e., the approximation is only valid for large masking ratio.

We will consider another approximation, but this time calculate the loss for a single sample. We consider a first-order approximation of f for a single sample.

$$f(x_R) \approx f(x_\mu) + \nabla f(x_\mu)(x_R - x_\mu),$$

which when substituted into the above loss gives us

$$\begin{aligned} \ell_m(x) =& \| x - f(x_\mu) \|^2 \\ &+ m(1-m) \operatorname{Tr}\left(H_x \operatorname{Blkdiag}(x^\top x) \right) \end{aligned}$$

Note that this is the loss for a single sample and F_x is the Fisher information matrix for data point x.

Masked autoencoders: A function space perspective Let us assume that we have access to the true input image signal $x(i,j)$, where $i, j \in [0,1]$, as opposed to a discretized version of it. The masked autoencoder objective can be posed as an optimization problem over functionals $f \in \mathcal{F}$, i.e.,

$$\ell_m = \int \| f(r \odot x) - x \|^2 \, \mathrm{d}r,$$

where r is a mask applied to the image. Assuming that f is linear, then the above objective reduces to

$$\ell_m = \|x - f(\mu)\|^2 - \|f(\mu)\|^2 + \int \|f(r \odot x)\|^2 \, \mathrm{d}r,$$

where $\mu = \int (r \odot x) \, \mathrm{d}r$. In the linear case, the MAE forces f to reconstruct the mean masked image, while minimizing the variance of the predictions made on the masked images.

A.2.3 Supporting lemmas

Lemma 1. *The loss of the masked autoencoder is*

$$\ell_m = \|X - (1-m)XAB\|^2 + m(1-m)\|GAB\|^2$$

Proof. Expanding the term inside the expectation, we get

$$\begin{aligned}
&\mathbb{E}_R \|X - (R \odot X)AB\|^2 \\
&\qquad = \mathbb{E}_R \mathrm{Tr}\,[X^\top X - 2X^\top (R \odot X)AB \\
&\qquad\qquad + B^\top A^\top (R \odot X)^\top (R \odot X)AB].
\end{aligned}$$

We note that $\mathbb{E}[\mathbb{R} \odot \mathbb{X}] = (1-m)$ and

$$\begin{aligned}
&\mathbb{E}[(R \odot X)^\top (R \odot X)] = \\
&\qquad \begin{cases} (1-m)X_i^\top X_j & X_i, X_j \text{ in the same patch} \\ (1-m)^2 X_i^\top X_j & X_i, X_j \text{ not in the same patch.} \end{cases} \\
&\qquad = (1-m)^2 X^\top X + m(1-m)\mathrm{Blkdiag}_p(X^\top X).
\end{aligned}$$

Substituting back into the masked autoencoder loss, we get

$$\begin{aligned}
&\mathbb{E}_R \| X - (R \odot X)AB \|^2 \\
&= \mathrm{Tr}\left[X^\top X - 2(1-m) X^\top XAB + (1-m)^2 X^\top X \right. \\
&\qquad \left. + m(1-m) B^\top A^\top \mathrm{Blkdiag}_p(X^\top X) AB \right] \\
&= \| X - (1-m)XAB \|^2 + m(1-m) \| GAB \|^2
\end{aligned}$$

where $G^\top G = \mathrm{Blkdiag}_p(X^\top X)$. □

Lemma 2. *For matrices $C \in \mathbb{R}^{d \times d}$, $X \in \mathbb{R}^{d \times k}$ and an invertible matrix $D \in \mathbb{R}^{d \times d}$, every critical point of*

$$L(X) = \mathrm{Tr}\left[(X^\top D X)^{-1} X^\top C X \right]$$

that is full-rank can be expressed as UQ, where Q is an invertible matrix and U is any subset of k eigenvectors of the generalized eigenvalue problem for (C, D).

Proof. Taking the derivative of $L(X)$ with respect to X and setting it to 0, we get

$$2DX(X^\top DX)^{-1} X^\top CX = 2CX.$$

Let (Λ_D, Φ_D) be the eigenvectors and eigenvalues of D. Let $(\Lambda_{\bar{C}}, \Phi_{\bar{C}})$, be the eigenvectors and eigenvalues $\Lambda_D^{-1/2} \Phi_D^\top C \Phi_D \Lambda_D^{-1/2}$. In addition, we define $\Phi = \Phi_{\bar{C}} \Lambda_D^{-1/2} \Phi_{\bar{D}}$ and $\tilde{X} = \Phi X$. We choose this definition of

Φ, since it diagonalizes both C and D.

$$
\begin{aligned}
&2DX(X^\top DX)^{-1}X^\top CX = 2CX \\
\Rightarrow &D\Phi^\top \tilde{X}(\tilde{X}^\top \Phi D\Phi^\top \tilde{X})^{-1}\tilde{X}^\top \Phi C\Phi^\top \tilde{X} = C\Phi^\top \tilde{X} \\
\Rightarrow &D\Phi^\top \tilde{X}(\tilde{X}^\top \tilde{X})^{-1}\tilde{X}^\top \Lambda_{\tilde{C}}\tilde{X} = C\Phi^\top \tilde{X} \\
\Rightarrow &\Phi D\Phi^\top \tilde{X}(\tilde{X}^\top \tilde{X})^{-1}\tilde{X}^\top \Lambda_{\tilde{C}}\tilde{X} = \Phi C\Phi^\top \tilde{X} \\
\Rightarrow &\tilde{X}(\tilde{X}^\top \tilde{X})^{-1}\tilde{X}^\top \Lambda_{\tilde{C}}\tilde{X} = \Lambda_{\tilde{C}}\tilde{X} \\
\Rightarrow &P_{\tilde{X}}\Lambda_{\tilde{C}}\tilde{X} = \Lambda_{\tilde{C}}P_{\tilde{X}}\tilde{X}
\end{aligned}
$$

where $P_{\tilde{X}}$ is the projection operator. Note that $P_{\tilde{X}}\tilde{X} = \tilde{X}$. Since $\Lambda_{\tilde{C}}$ is diagonal, $P_{\tilde{X}}$ must also be diagonal in order for the matrices to commute. Furthermore, $P_{\tilde{X}}$ has exactly k eigenvalues equal to 1 and the rest set to 0, since X has rank k. Hence, $\tilde{X}$ must be of the form $I_{S_k}Q$ where S_k selects a subset of k dimensions and $Q \in \mathbb{R}^{k\times k}$ is an invertible matrix. Hence $X = \Phi_{S_k}Q$ where Φ_{S_k} is a subset of k eigenvectors of the generalized eigenvalue problem. □

Lemma 3. *For matrices $C \in \mathbb{R}^{d\times d}$, $X \in \mathbb{R}^{d\times k}$ and an invertible matrix $D \in \mathbb{R}^{d\times d}$, the global maximum of*

$$L(X) = \mathrm{Tr}\left[(X^\top DX)^{-1}X^\top CX\right]$$

is $\sum_{i=1}^{k}\Lambda_k$ where Λ are the eigenvalues of the generalized eigenvalue problem (C, D).

Proof. From lemma 2, we know that any critical point is of the form $\Phi_{S_k}Q$. Subtituting this into $L(X)$, we get

$$
\begin{aligned}
L(X) &= \mathrm{Tr}\left[(X^\top DX)^{-1}X^\top CX\right] \\
&= \mathrm{Tr}\left[(Q^\top Q)^{-1}Q^\top \Phi_{S_k}^\top C\Phi_{S_k}Q\right] \\
&= \mathrm{Tr}\left[\Phi_{S_k}^\top C\Phi_{S_k}\right] = \sum_{i\in S_k}\Lambda_i.
\end{aligned}
$$

The loss is maximized by the largest k eigenvalues and minimized by the smallest k eigenvalues.

□

Lemma 4. *Under the Taylor series approximation of* $f(X_R) \approx f(0) + X_R \nabla f(0)^\top = X_R \nabla f(0)^\top$*, the MAE loss for a non-linear function* f *is*

$$\ell_m = \| X - (1-m) X \nabla f(0)^\top \|^2 + m(1-m) \| G \nabla f(0)^\top \|^2.$$

Proof.

$$\begin{aligned}
\ell_m &= \| X - X_R \nabla f(0)^\top \|^2 \\
&= \| X \|^2 + \mathbb{E}_R \| X_R \nabla f(0)^\top \|^2 - 2\mathbb{E}_R \operatorname{Tr}\left(X^\top X_R \nabla f(0)\right) \\
&= \operatorname{Tr}\left(X^\top X\right) + (1-m)^2 \operatorname{Tr}\left(\nabla f(0) X^\top X \nabla f(0)^\top\right) \\
&\qquad + m(1-m) \operatorname{Tr}\left(\nabla f(0) \text{Blkdiag}(X^\top X) \nabla f(0)^\top\right) \\
&\qquad - 2(1-m) \mathbb{E}_R \left[X^\top X \nabla f(0)^\top\right] \\
&= \| X - (1-m) X \nabla f(0)^\top \|^2 + m(1-m) \| G \nabla f(0)^\top \|^2.
\end{aligned}$$

□

Lemma 5. *Under the Taylor series approximation of* $f(x_R) \approx f(x_\mu) + \nabla f(x_\mu)(x_R - x_\mu)$*, the MAE loss, reduces to*

$$\ell_m(x) = \| x - f(x_\mu) \|^2 + m(1-m) \operatorname{Tr}\left(H_x \text{Blkdiag}(x^\top x)\right).$$

Proof.

$$\begin{aligned}
\ell_m(x) &= \| x - f(x_\mu) - \nabla f(x_\mu)(x_R - x_\mu)\|^2 \\
&= \| x - f(x_\mu)\|^2 + \mathbb{E}_R\|\nabla f(x_\mu)(x_R - x_\mu)\|^2 \\
&\quad + 2\mathbb{E}_R(x - f(x_\mu))^\top(\nabla f(x_\mu)(x_R - x_\mu)) \\
&= \| x - f(x_\mu)\|^2 + \\
&\quad \mathbb{E}_R(x_R - x_\mu)^\top \nabla f(x_\mu)^\top \nabla f(x_\mu)(x_R - x_\mu) \\
&= \| x - f(x_\mu)\|^2 + \\
&\quad \mathbb{E}_R \operatorname{Tr}\left(\nabla f(x_\mu)^\top \nabla f(x_\mu)(x_R - x_\mu)(x_R - x_\mu)^\top\right) \\
&= \| x - f(x_\mu)\|^2 + m(1-m)\operatorname{Tr}\left(F_x \operatorname{Blkdiag}(x^\top x)\right).
\end{aligned}$$

□

Lemma 6. *The masked autoencoder loss, for a input image x and linear functional f is*

$$\ell_m = \| x - f(\mu)\|^2 - \| f(\mu)\|^2 + \int \| f(r \odot x)\|^2 \, \mathrm{d}r,$$

where $\mu = \int (r \odot x) \, \mathrm{d}r$

Proof.

$$\begin{aligned}
\ell_m &= \int \| x - f(r \odot x)\|^2 \, \mathrm{d}r \\
&= \int \| x - f(\mu) - (f(r \odot x) - f(\mu))\|^2 \, \mathrm{d}r \\
&= \int \| f(\mu) - x\|^2 + \| f(r \odot x) - f(\mu)\|^2 \, \mathrm{d}r \\
&\quad - \int 2\langle x - f(\mu), f(r \odot x) - f(\mu)\rangle \mathrm{d}r \\
&= \int \| f(\mu) - x\|^2 + \| f(r \odot x) - f(\mu)\|^2 \, \mathrm{d}r \\
&= \| x - f(\mu)\|^2 - \| f(\mu)\|^2 + \int \| f(r \odot x)\|^2 \, \mathrm{d}r.
\end{aligned}$$

□

A.3 Additional experiments for Model Zoo

A.3.1 Understanding task competition

To understand which tasks aid each other's learning and which compete for capacity and may thereby deteriorate performance, we investigated the Coarse-CIFAR100 dataset extensively. We first computed the pairwise task competition by comparing the relative gain/drop in classification accuracy of each pair of tasks when the row task is trained in isolated versus training the row and column tasks together using a simple multi-task learner (Multi-Head). fig. A.4 discusses the results.

fig. A.5, is the extended version of fig. 2.1. It shows the validation accuracy of each task (along a single row) as more tasks are added to Multi-Head. Each column is a single Multi-Head model trained on a subset of tasks from scratch. As more tasks are added, the accuracy of most tasks increase However, the increase is not monotonic with each added task, and if one follows a particular row, there are non-trivial patterns wherein adding a particular task may deteriorate the performance on the row task and adding some other task later may recover the lost accuracy. This is a direct demonstration of the tussle between the task competition term (first) and the concentration term (third) in theorem 5. This indicates that training on the appropriate set of tasks is crucial to learn from multiple tasks.

Next, we investigated such task competition on other continual learning datasets, namely, Permuted-MNIST, Rot-MNIST, Split-CIFAR10, and Split-MNIST. It is clear from fig. A.6 that there is very little competition in this case. Either the tasks are quite different from each other (like the case of Permuted-MNIST), or they are synergistic (most cells are green), or they do not hurt each other's performance, i.e., they may correspond to our model of task relatedness.

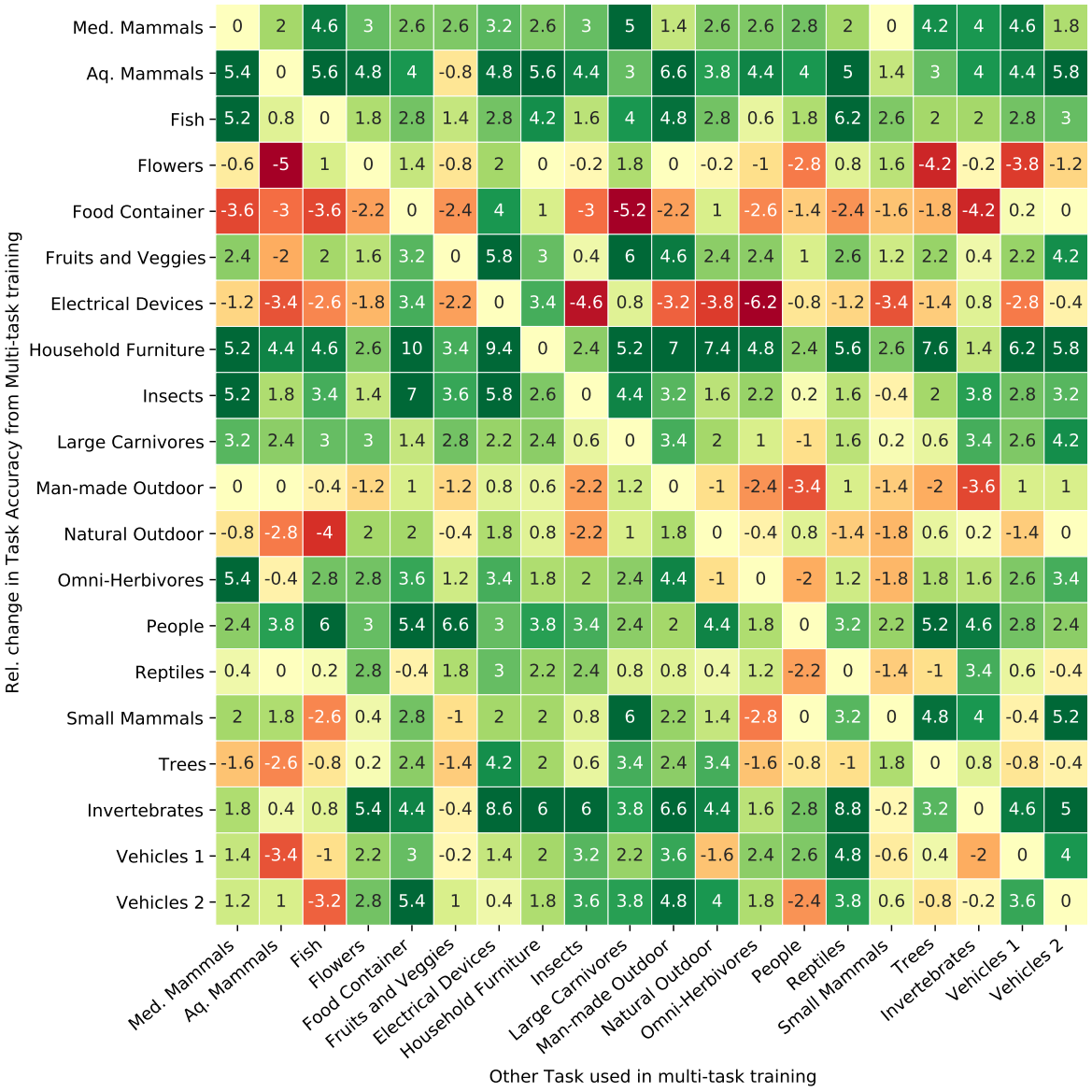


Figure A.4: Pairwise task competition matrix. Cells are colored by the gain(green)/loss(warm) of accuracy of pairwise Multi-Head training as compared to training the row-task in isolation; this is a good proxy for the transfer coefficient ρ_{ij} in eq. (2.3). Although most pairs benefit each other (green), certain tasks, e.g., "Food Container" are best trained in isolation while others such as "Aquatic Mammals" are typically detrimental to most other tasks. One can study this matrix and identify many more such properties. In summary, whether tasks aid or hurt each other is quite nuanced even for CIFAR100.

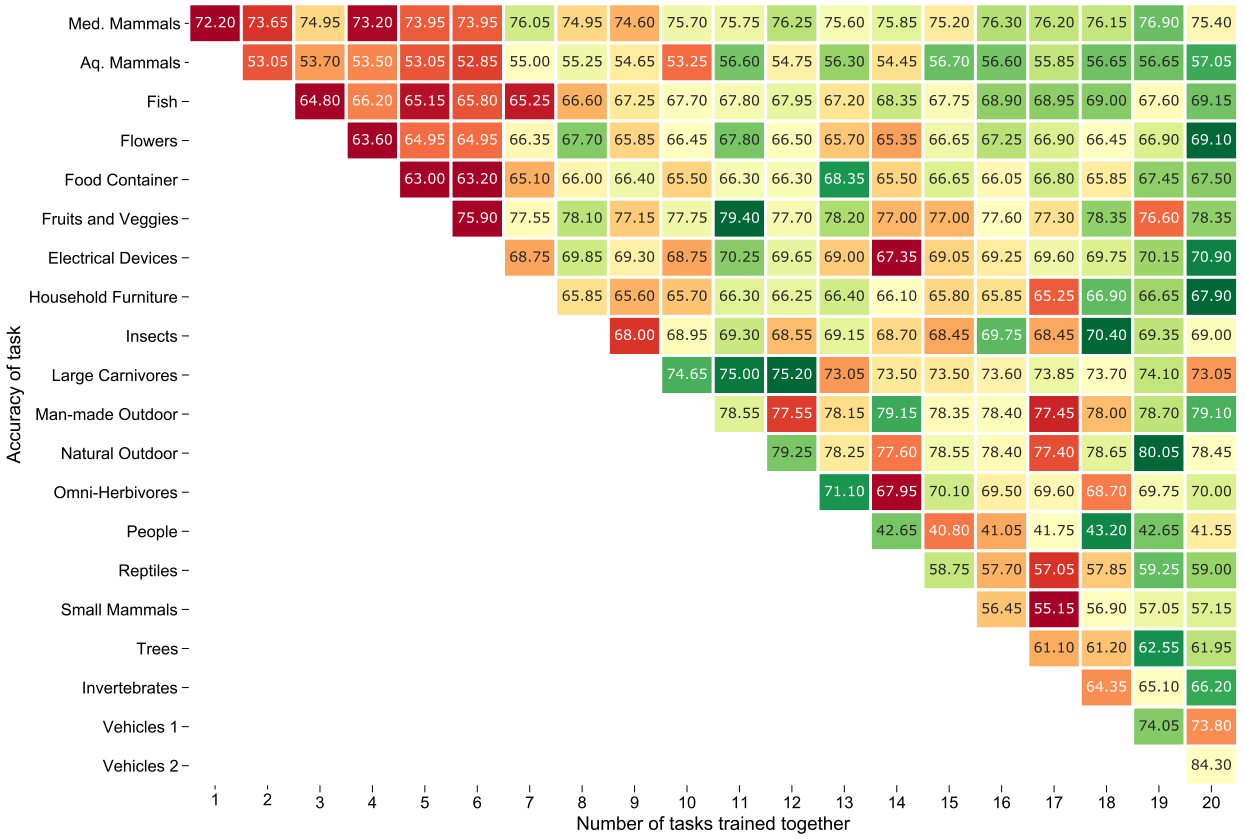


Figure A.5: In order to demonstrate how some tasks help and some tasks hurt each other, we train a number of multi-task learners for a varying number of tasks (X-axis) and track the accuracy on each of the tasks from Coarse-CIFAR100 (100 samples/label for each task). The order of tasks is the same for rows (top to bottom) and the columns (left to right). In other words, the first cell (the diagonal) indicates the accuracy of the task trained by itself in isolation (Isolated). Cells are colored warm if accuracy is worse than the median accuracy of that row. For instance, multi-task training with 11 tasks is beneficial for "Man-made Outdoor" but accuracy drops drastically upon introducing task #12, it improves upon introducing #14, while task #17 again leads to a drop. One may study the other rows to reach a similar conclusion: there is non-trivial competition between tasks, even in commonly used datasets. Tackling this issue effectively is the key to obtaining good performance on multi-task learning problems

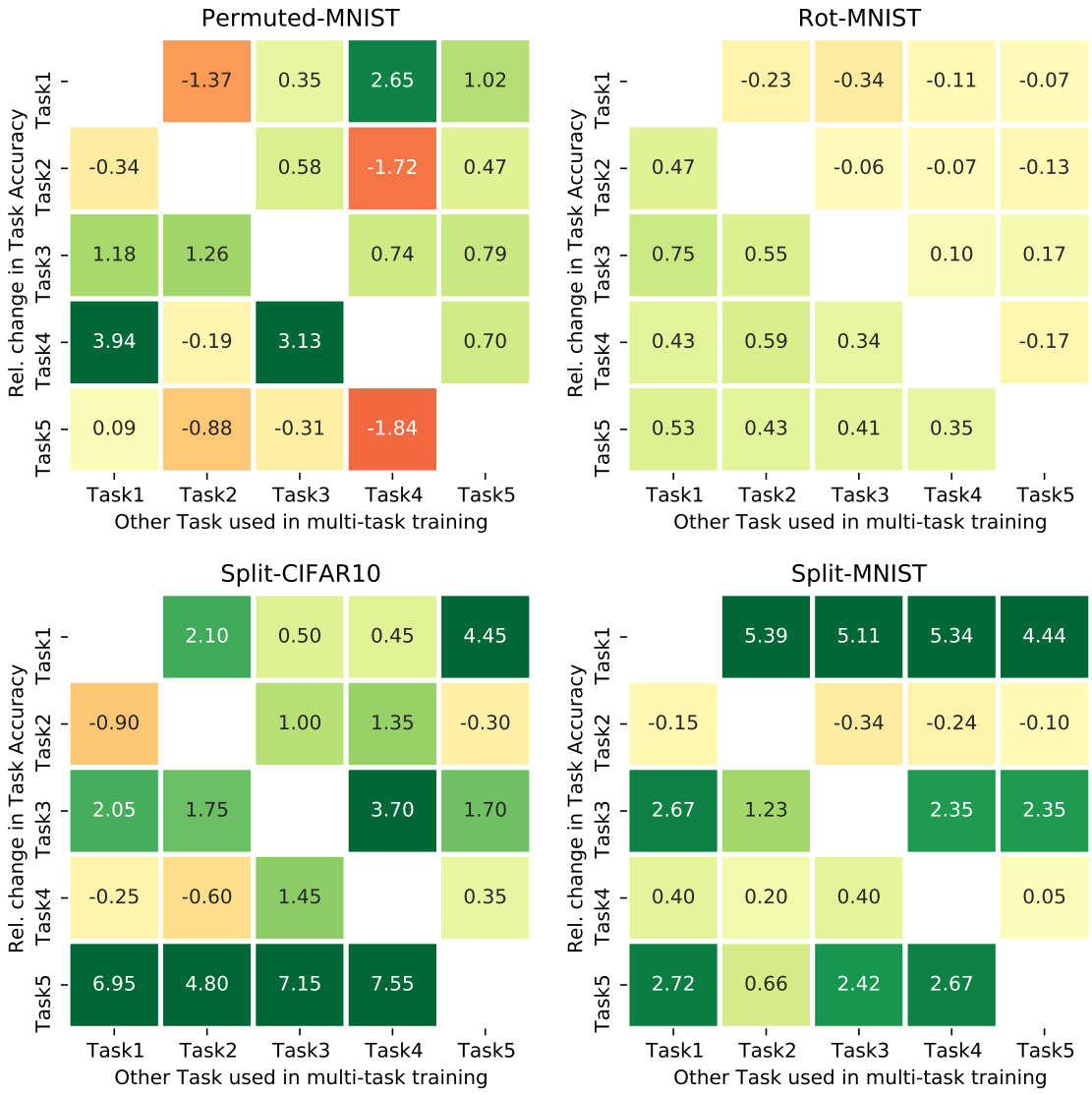


Figure A.6: Each row is the relative increase/decrease (green/red) in accuracy of a two task multi-task learner compared to training on the task corresponding to the particular row in isolation; all entries are computed using 100 samples/class. Cells are colored green for accuracy gained, and warm for accuracy dropped; the entries in this matrix are a good proxy for the transfer coefficient ρ_{ij} in eq. (2.3). A similar plot for Coarse-CIFAR100 tasks is shown in the right panel of fig. 2.1. Split-CIFAR10 and Split-MNIST indicate that most tasks mutually benefit each other. This is also true, but to a lesser extent, for Rotated-MNIST. Permuted-MNIST is a qualitatively different problem than these, perhaps because there is no obvious relationship between the tasks and there exist some tasks that lead to a large deterioration of accuracy.

A.3.2 Visualizing successive iterations of Model Zoo

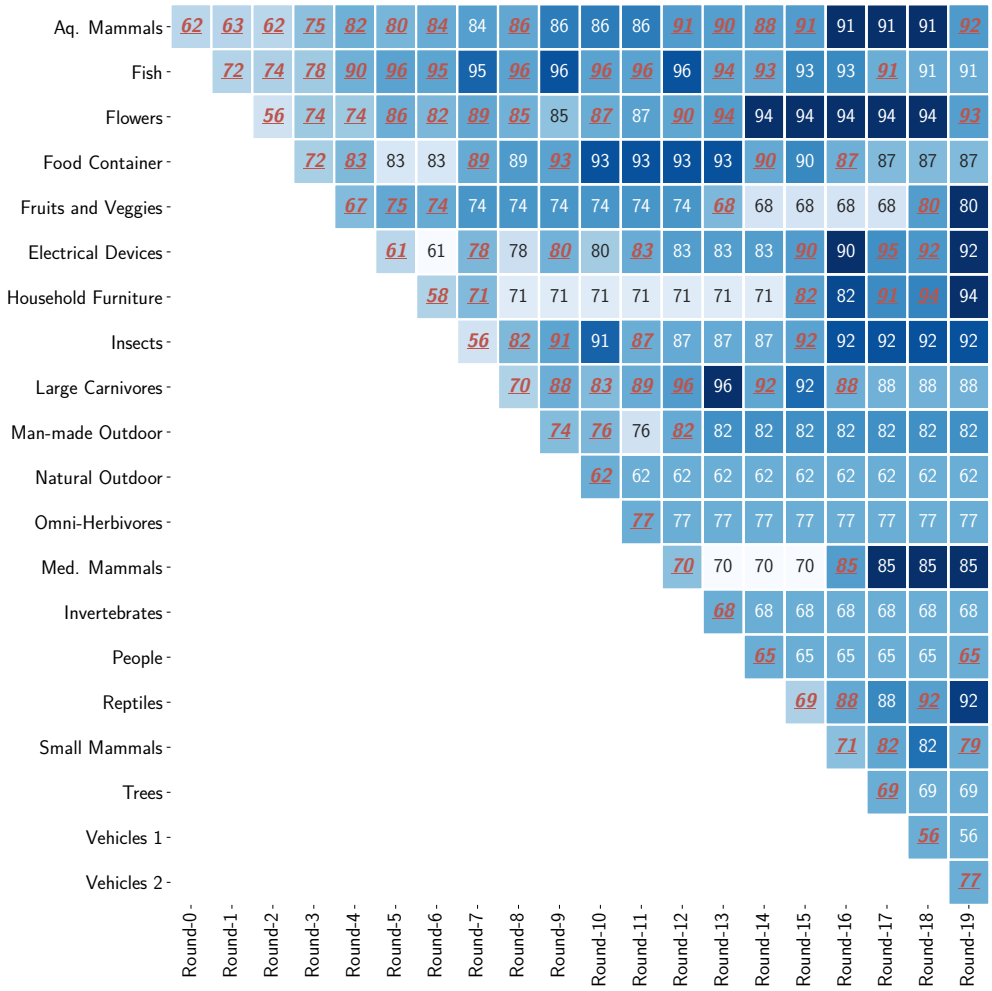


Figure A.7: The iterations of Model Zoo are visualized for the Split-miniImagenet dataset for 20 rounds, with 5 tasks selected in every iteration of Model Zoo. Red elements are tasks that were selected by boosting in that particular round. We observe that the accuracy of most tasks improves over the rounds, which indicates the utility of Model Zoo-like training scheme This plot also indicates that Model Zoo can improve the per-task accuracy on nearly all tasks. The model is trained for only a single-epch per boosting round.

In order to understand how the accuracy of Model Zoo evolves on all tasks as a function of the episodes, we created fig. A.7. This is a very insightful picture and we can draw the following conclusions from it.

(i) The accuracy along the diagonal of most tasks increases along the row, i.e., across episodes. Only for a few tasks like Food Container, the accuracy drops in later episodes. Note that we also see from fig. A.4 that Food Container is a task that is best trained in isolation because it leads to deterioration of accuracy

when trained with essentially any other task.

(ii) There is strong backward transfer throughout the dataset, i.e., the accuracy of a task shown in earlier rounds increases, as later synergistic tasks are shown to the learner.

(iii) We also see strong forward transfer. Roughly speaking, in the second half of the rows, the tasks already have a good initial accuracy.

We advocate that such plots should be made for different continual learning algorithms to obtain a precise picture of the amount of forward and backward transfer.

A.3.3 Single Epoch Metrics

We obtain metrics from publicly available implementations of a few different continual learning algorithms, which are shown in tables A.1 and A.2. We see that Model Zoo and its variants uniformly have essentially no forgetting and good forward transfer. The average per-task accuracy is also higher than existing methods on these datasets. These tables show results for single-epoch training (to be consistent with the implementation of these existing methods).

Method	Avg. Accuracy	Forgetting	Forward
SGD	34.52	19.88	53.30
EWC	34.71	18.60	52.19
AGEM	37.23	16.96	52.72
ER	41.36	14.29	54.87
Stable-SGD	37.27	12.07	48.43
TAG	43.33	12.39	55.1
Isolated-small	58.719	0.0	58.71
Model Zoo-small	60.3	0.370	59.13
Isolated-large	41.28	0.0	41.28
Model Zoo-large	46.98	0.38	44.43

Table A.1: Single Epoch continual learning metrics on Coarse-CIFAR100

A.3.4 Tracking Individual Task Accuracies

We next study how the individual per-task accuracy evolves on different datasets. The following figures are extended versions of the right panel of fig. 2.3. We see that the accuracy of all tasks increases with successive episodes. This is quite uncommon for continual learning methods and indicates that Model Zoo essentially does not suffer from catastrophic forgetting. We have also juxtaposed the corresponding curves of the single-epoch setting with the multi-epoch training in Model Zoo; we would like to demonstrate the dramatic gap

Method	Avg. Accuracy	Forgetting	Forward
SGD	46.69	16.653	62.35
EWC	47.93	14.26	61.34
AGEM	51.86	10.102	61.13
ER	55.41	9.52	64.03
Stable-SGD	49.28	9.76	57.79
TAG	58.38	5.15	63.00
Isolated-small	65.8	0.0	65.8
Model Zoo-small	81.049	1.278	66.57
Isolated-large	40.2	0.0	40.25
Model Zoo-large	64.12	0.27	48.34

Table A.2: Single Epoch continual learning metrics on Split-MinImagenet

in the accuracy of these problem settings. Even if single-epoch variant of Model Zoo also does not forget (its accuracy is much better than existing continual learning methods), the multi-epoch variant has much higher accuracy for every task. This indicates that continual learning algorithms should also focus on per-task accuracy in addition to mitigating forgetting, if they are to be performant. The performance of Model Zoo is evidence that we can build effective continual learning methods that do not forget.

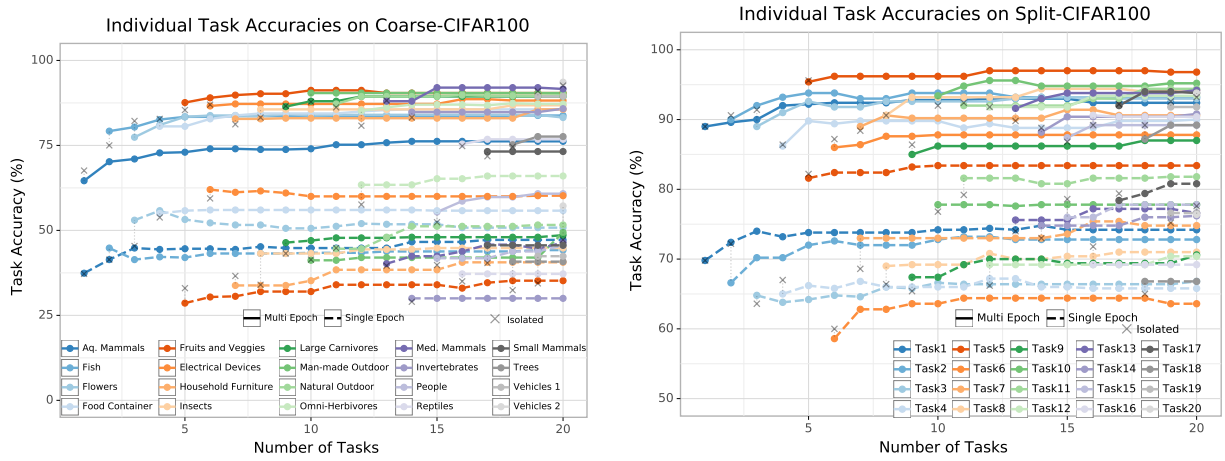


Figure A.8: Evolution of task accuracy on Coarse-CIFAR100 and Split-CIFAR100

A.3.5 Additional Continual Learning Experiments on 100 samples/label

We also performed continual learning experiments with 100 samples/class in table A.3. We find that Model Zoo obtains an accuracy that lies in between those of Isolated and the approximate upper bound given by

Multi-Head (multi-task learning). Doing so indicates strong ability of the learner for *both* forward and backward transfer. In some cases, the continual learner even outperforms Multi-Head trained on all tasks together.

Dataset	Isolated	Multi-Head (multi-task)	Model Zoo
Rotated-MNIST	98.17 ± 0.24	98.47 ± 0.18	98.44 ± 0.17
Split-MNIST	97.11 ± 1.21	99.47 ± 0.08	98.98 ± 0.51
Permuted-MNIST	84.59 ± 1.65	86.36 ± 1.15	86.04 ± 1.68
Split-CIFAR10	82.09 ± 0.76	85.73 ± 0.60	84.17 ± 0.60
Split-CIFAR100	80.04 ± 0.44	87.93 ± 0.50	86.27 ± 0.19
Coarse-CIFAR100	65.34 ± 0.41	69.05 ± 0.38	66.80 ± 6.27

Table A.3: Average per-task accuracy (%) at the end of all episodes using 100 samples/class, bootstrapped across 5 datasets (mean ± std. dev.). Model Zoo performs better than Isolated on all problems even if tasks are shown sequentially.

We next visualize the evolution of the per-task test accuracy for various datasets in fig. A.9. This is a qualitative way to investigate forward and backward transfer in the learner. Forward transfer is positive if the accuracy of a newly introduced task in a particular episode is higher than what it would be if the task were trained in isolation. Backward transfer is positive if successive episodes and tasks result in an increase in the accuracy of tasks that were introduced earlier in continual learning. Both section A.3.4 and fig. A.9 consistently show non-trivial forward and backward transfer.

A.4 Additional experiments for redundancy in perception tasks

A.4.1 Projections of images onto different subspaces

We project an image of CIFAR-10 onto different bands of Eigenvectors, frequencies and wavelets. Surprisingly, most of these bands have

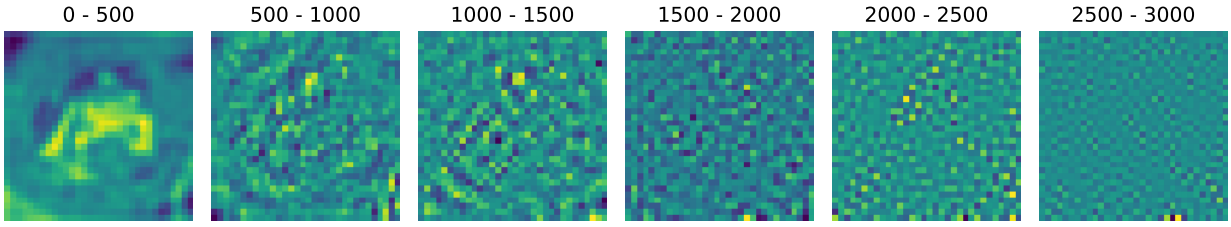


Figure A.10: Average CIFAR-10 image when projected into different PCA subspaces.

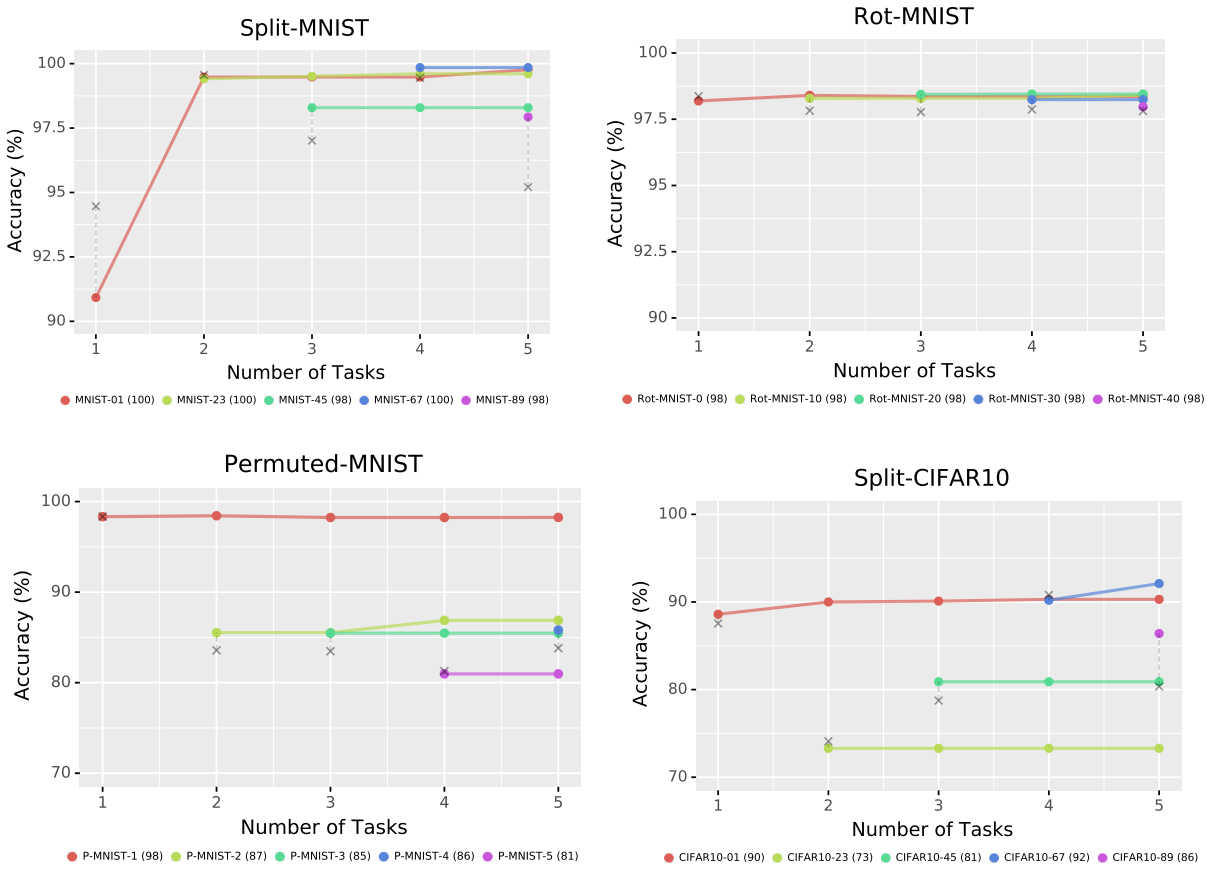


Figure A.9: Per-task validation accuracy as a function of the number of episodes of continual learning for problems using variants of CIFAR10 and MNIST datasets using Model Zoo. Each task has 100 samples/class. X-markers denote accuracy of Isolated on the new task. We see both forward transfer (Model Zoo often starts with a higher accuracy than Isolated) and backward transfer (accuracy of some past tasks improves in later episodes). For problems like Permuted-MNIST and Rotated-MNIST, there is little forward or backward transfer.

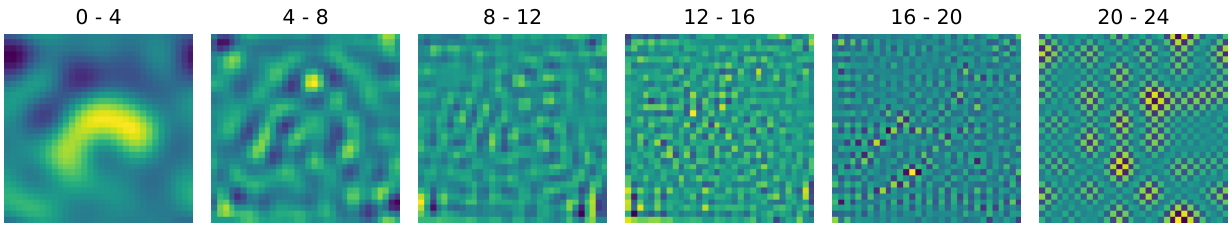


Figure A.11: Average CIFAR-10 image when projected into different radial frequency bands.

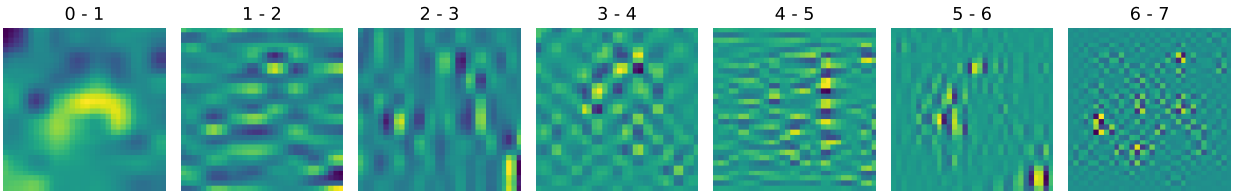


Figure A.12: Average CIFAR-10 image when projected into different wavelet scale bands.

A.4.2 Partial Information Decomposition

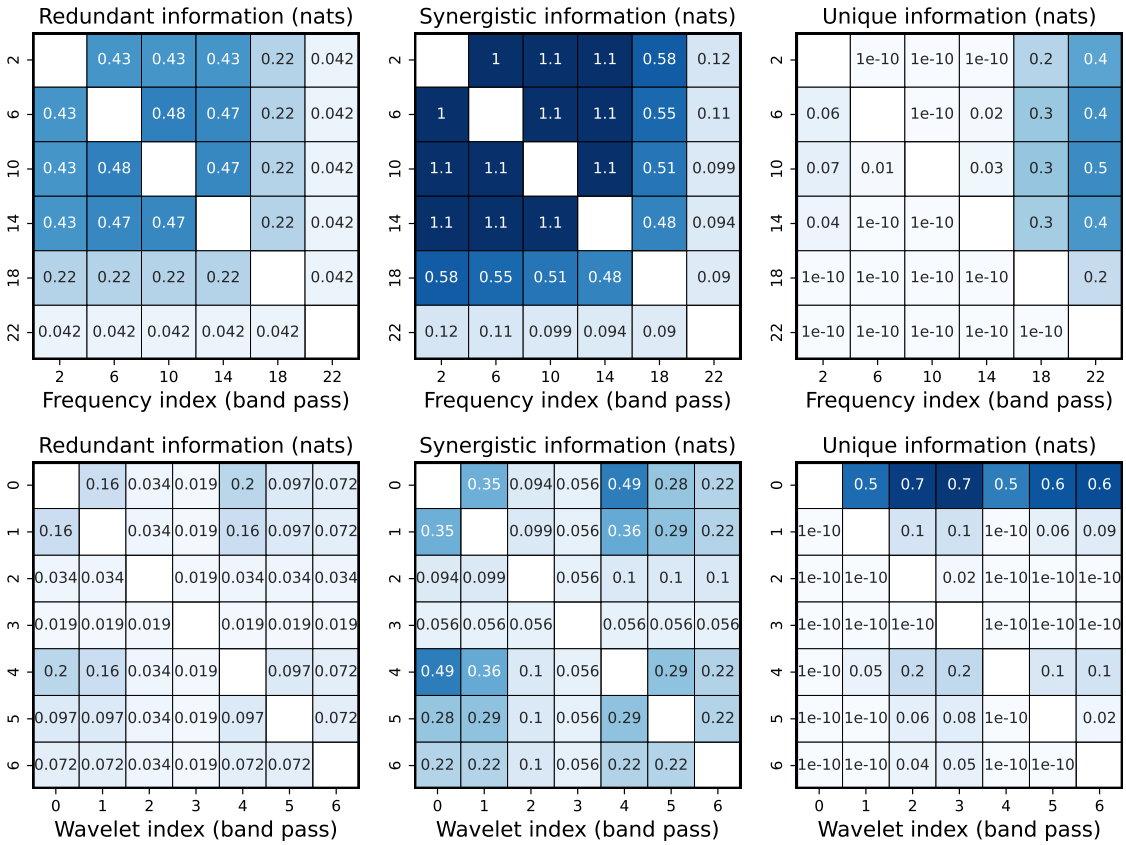


Figure A.13: Different bands have high amounts of redundant and synergistic information even with frequencies and wavelets for CIFAR-10. This corroborates the result in fig. 4.17. It also suggests that our technique for calculating mutual information is reliable; we see that redundant and synergistic information are large while unique information is small. Note that the numbers here are in log-scale, while the ones in the main paper are in nats.

A.4.3 Decay in power or spectral density

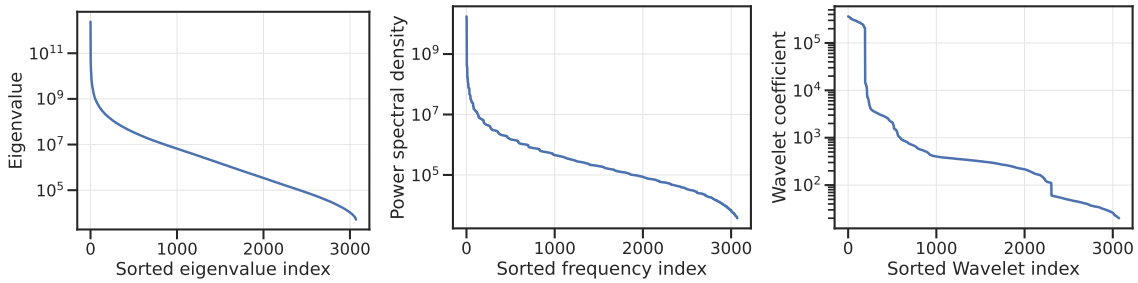


Figure A.14: The decay in coefficients of the Eigenvalues frequencies and wavelets. The spectra of all 3 linear bases decay exponentially and have characteristic small head and a long tail.

A.4.4 Understanding redundancy in feature space

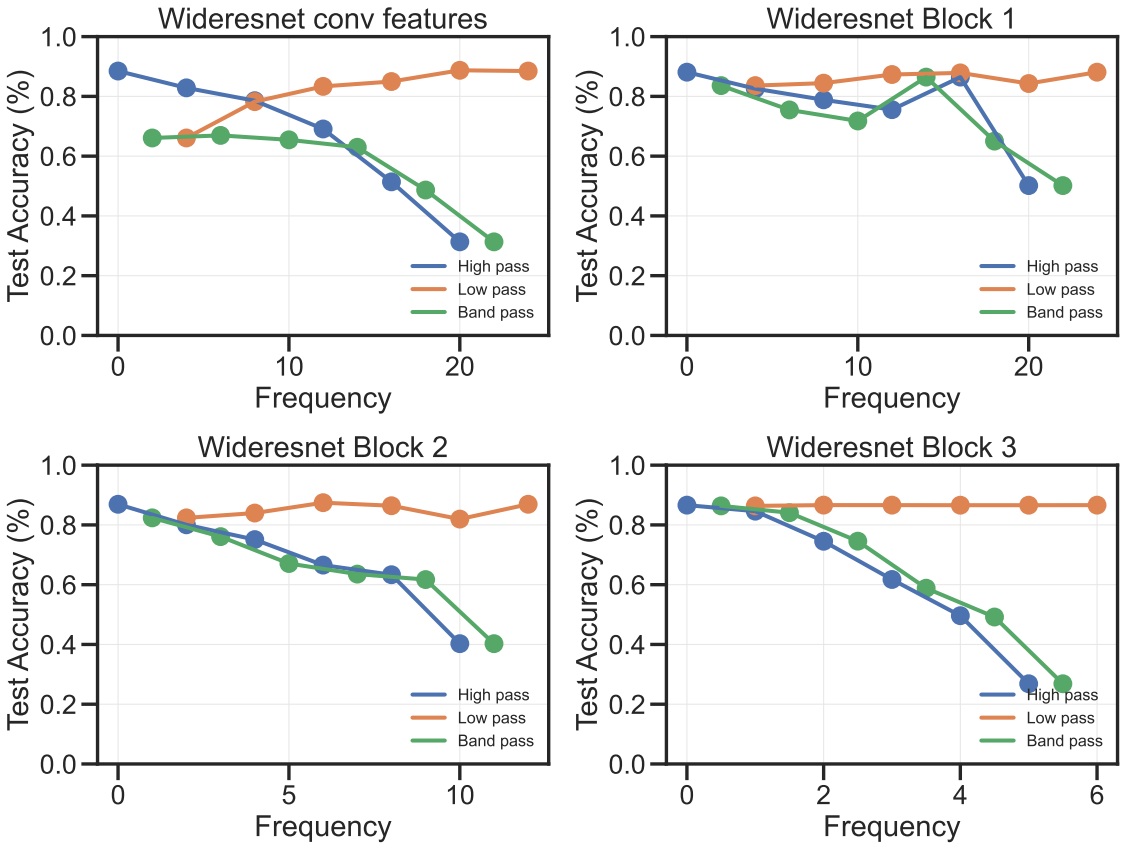


Figure A.15: There is also redundancy in the learned features of a trained network. The experimental setup of this figure is identical to that of fig. 4.18 (top) except that instead of creating subspaces using PCA, Fourier and wavelet bases computed from raw pixels, we create these bands using the features from different intermediate layers of the trained network. In other words, there is redundancy in the original input data because many subspaces are predictive of the task. But the network does not completely remove this redundancy when it learns the features.

A.4.5 Understanding bands of eigen-vectors, Fourier and Wavelet coefficients

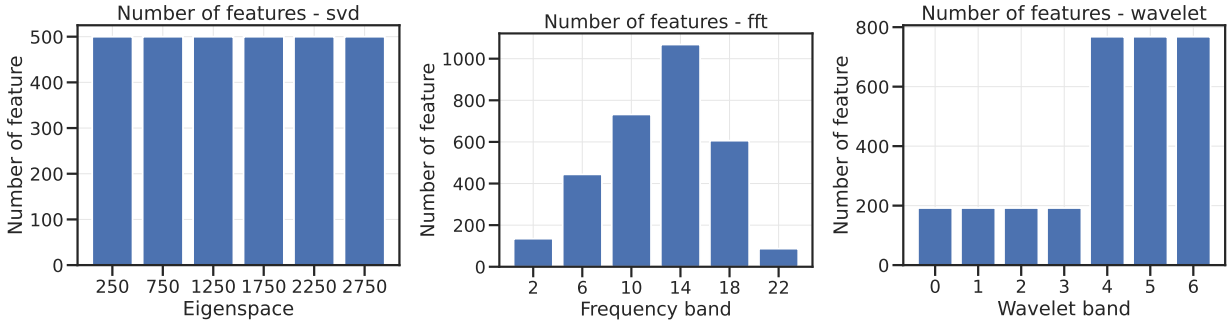


Figure A.16: Number of features in different bands of the input.

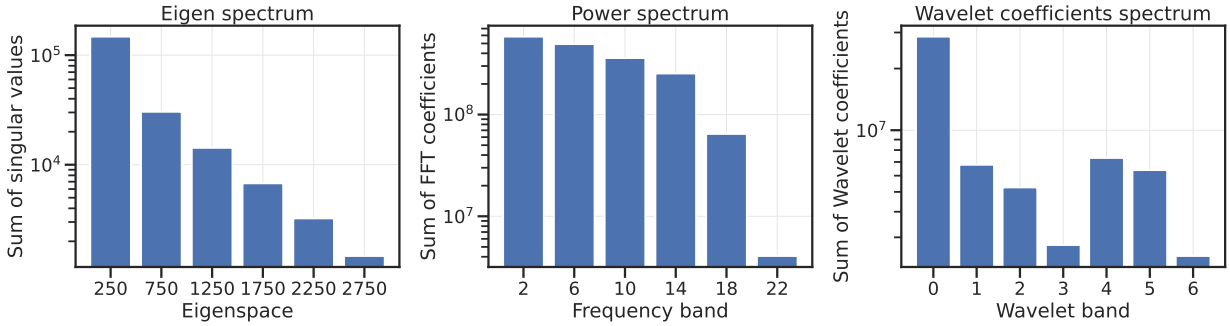


Figure A.17: Sum of singular values, Fourier and wavelet coefficients in different bands.

A.4.6 Trained Network Frequency Sensitivity, Additional Experiments

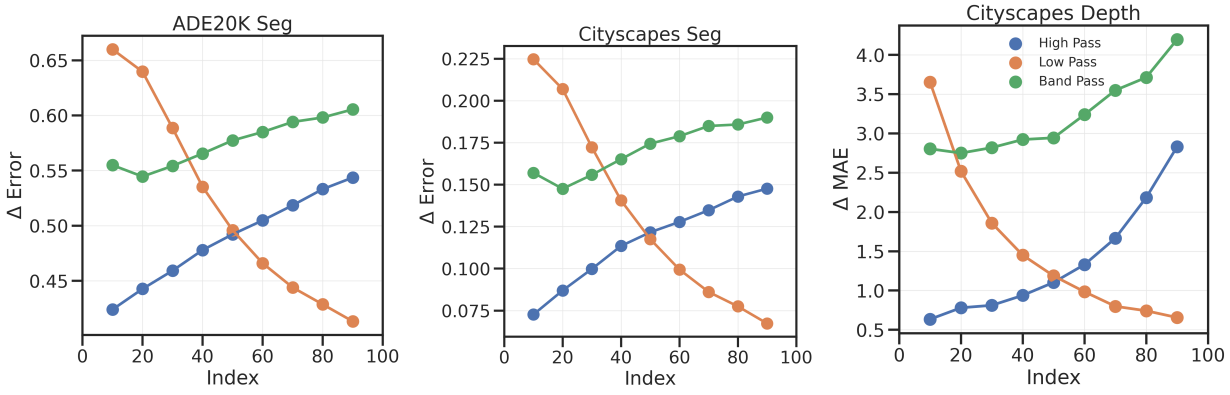


Figure A.18: We showcase some additional experiments of taking a pretrained network and applying various filters for inference to the network and observing network performance. We find a similar analysis to other experiments in the paper showcasing a bias towards using components of the low frequency prediction.

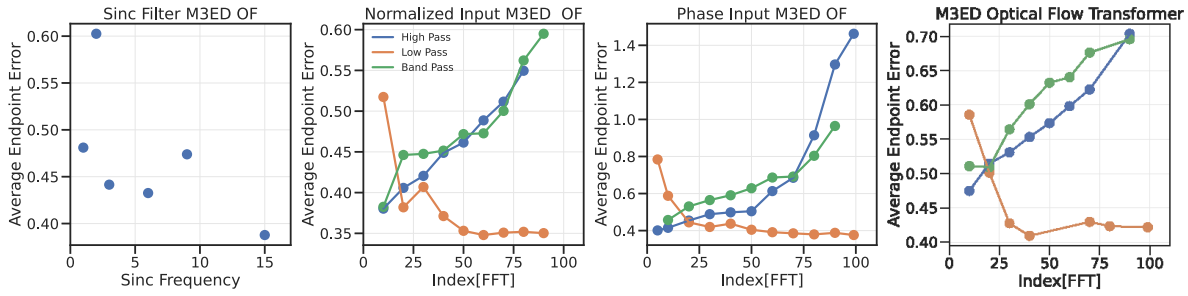


Figure A.19: In addition to the standard set of baselines shown we also created some additional results on the M3ED optical flow results. 1) Normalize the input after filtering to range from -1-1. 2) Create a mask just on the phase distribution of the FFT(note this mask was a hard 0-1, mask as opposed to Butterworth filter), 3) Sinc filter that nulls out bands according to sinc frequency. 4) Efficient Former V2 (Li et al., 2023) was used for the standard high Pass, low pass and band pass in this work.

A.4.7 Additional Perception Filter Experiments

In addition to the band-pass, low-pass, and high-pass filtering experiments in the main work, we conducted further experiments to investigate the impact of these factors on task performance.

The first experiment involved applying a sinc mask to the frequency distribution of the original input data. This experiment was motivated by our observation that masked autoencoders manipulate the frequency domain in this way. Since the size of the mask influences the sinc frequency, we wanted to analyze the resulting effects. We found that increasing the sinc frequency led to a consistent decrease in the average endpoint error for optical flow estimation.

The next experiment examined the effect of normalizing the input distribution to a range of 0-1 across all our experiments. We observed that the trends remained largely unchanged compared to using the unnormalized input. This is likely due to the batch normalization layers employed in the U-Net architecture.

In another experiment, we investigated masking out the phase information of the original input image instead of the amplitude in the frequency spectrum. This approach was inspired by research in psychophysics and signal processing (Piotrowski and Campbell, 1982; Thomson et al., 2000; Wichmann et al., 2006), which highlights the impact of phase statistics on image perception. The results of this experiment mirrored those of the original M3ED experiments, showing that performance improves with a larger portion of the spectrum preserved and that low-frequency information leads to better performance compared to high-frequency information.

For our final set of experiments, we aimed to investigate changes in performance by replacing the U-Net architecture with the EfficientFormer V2 model (Li et al., 2023). The overall trends in the results remained consistent with those obtained using the U-Net model.

A.5 Additional experiments with masked autoencoders

Additional details about MAE pretraining We train MAEs using the architecture in He et al. (2021). We divide the image into patches of size p and randomly mask a fraction m of the patches before feeding it to the MAE. The encoder projects the unmasked patches to a d-dimensional embedding using a linear layer. The sequence of patches are then fed to a series of Transformer blocks. The decoder adds a learnable vector and position encoding for every masked patch and reconstructs the masked patches.

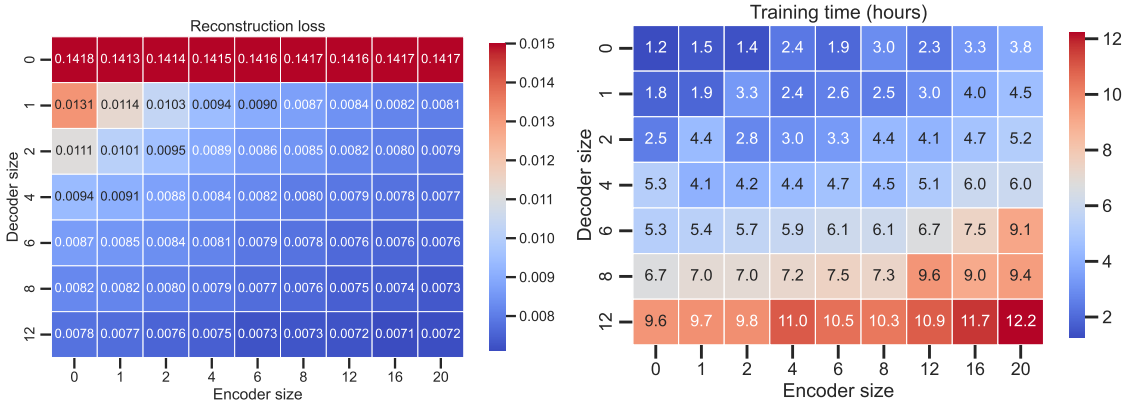


Figure A.20: We vary the number of encoder and decoder layers and record the reconstruction loss at the end of training and the time required the train the MAEs. Note that **MAEs are slow to train** with training time growing faster the size of the decoder. The reconstruction loss becomes smaller with increasing size of both the decoder and the encoder. However, we find that the **training loss is not a good proxy for downstream task performance**.

Number of Encoder and decoder layers We train MAEs on CIFAR10 for different encoder and decoder sizes. We find that the reconstruction loss decreases with increasing size of both the encoder and the decoder (fig. A.20). However, the training time grows faster than the size of the decoder, making it computationally expensive to train large decoders. We also find that training loss is not a good proxy for downstream task performance. We evaluate the performance of the trained encoder using linear probing and find that the accuracy improves as we increase the size of the encoder. However, the optimal decoder size is 2-4 layers (fig. A.21).

If the model are trained only using the supervised loss, i.e., we do no MAE pretraining, then the accuracy on

CIFAR plateaus around 83-84%. In fact the accuracy for a 20-layer network is worse than the accuracy for a 12-layer network which differs from the trend for masked autoencoders.

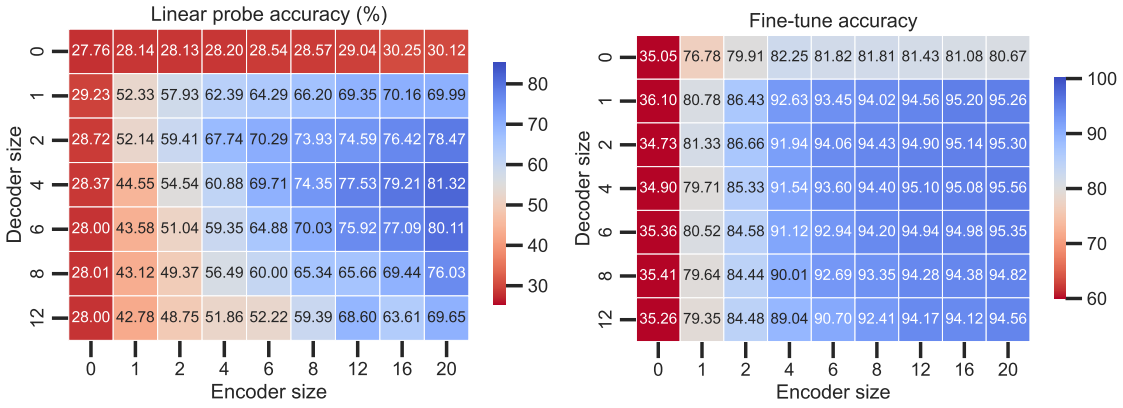


Figure A.21: We vary the number of encoder and decoder layers (transformer blocks) and plot the linear probe accuracy (left) and the accuracy after fine-tuning for 100 epochs. The **accuracy of the trained encoder continues to improve as we increase its size**. Linear probe accuracies are usually indicative of performance after fine-tuning.

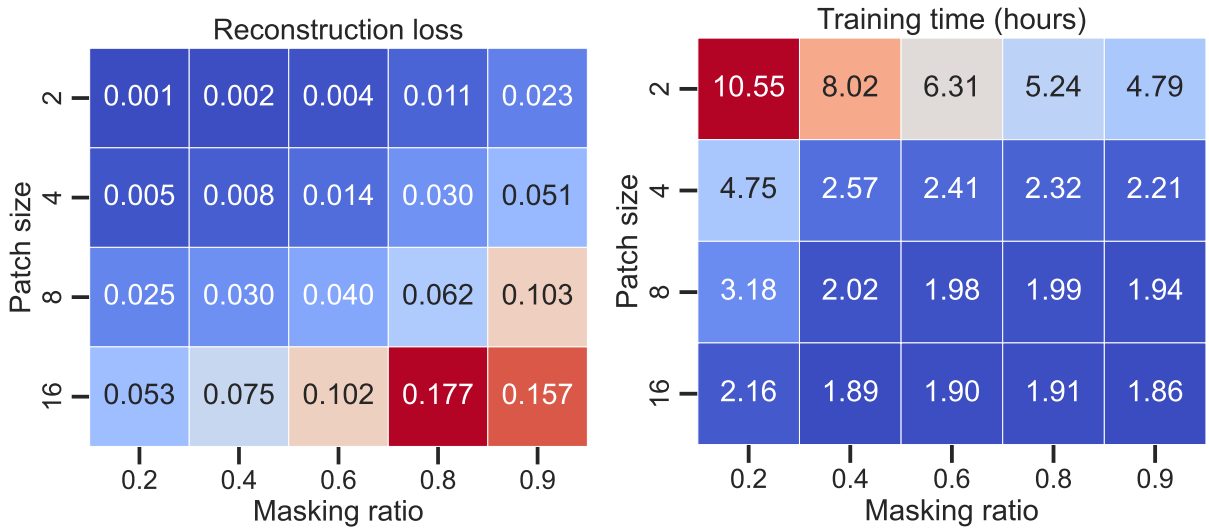
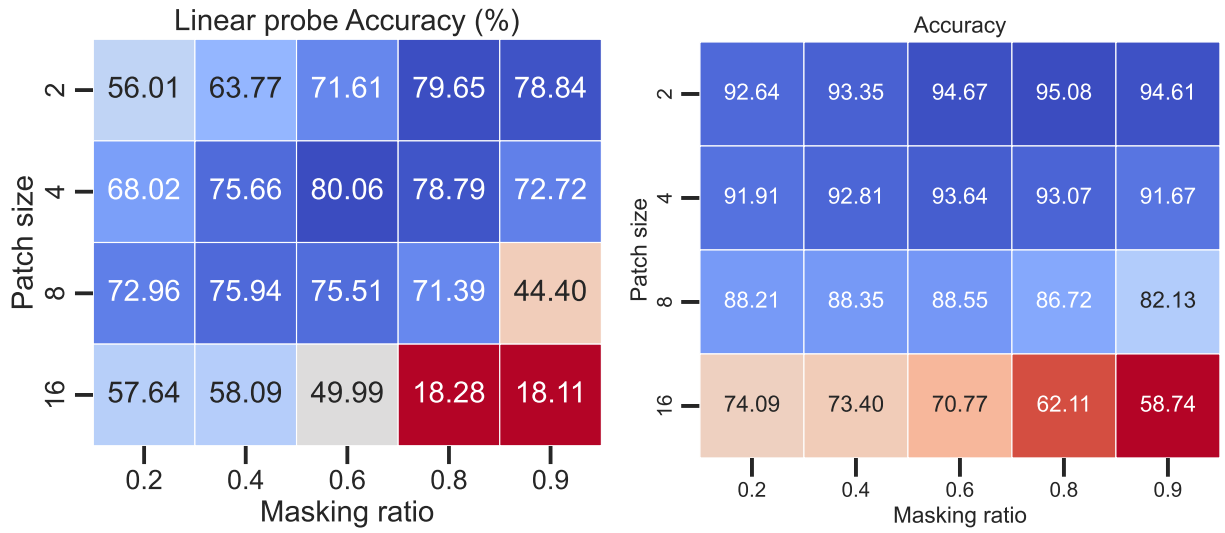


Figure A.22: We vary the masking ratio and patch-size of the masked autoencoder. While larger masking ratio lead to smaller training times, even smaller masking ratios work quite well. Smaller patch-sizes are a lot slower to train but usually perform better than larger patch-sizes.

Patch-size vs. Masking ratio The masking ratio and patchs-size control the basis learnt by the masked autoencoder. We surprisingly find that many different parameters work surprisingly well for downstream task accuracy. We also note that reconstruction loss is not indicative of downstream task performance.

Are long training times even necessary? MAEs are typically trained for a large number of epochs and the reconstruction error continues to decrease over the course of training. However, the reconstruction error is not



[Training time as a function of masking-ratio and patch-size]

Figure A.23: We plot the (left) linear probe accuracy and the accuracy after fine-tuning (right) for different patch-size and masking ratio.

predictive of both the linear-probe and fine-tuning accuracies. In fig. 3.12, we consider multiple checkpoints over the course of pretraining and plot the number of pretraining epochs against the linear probe accuracy of that checkpoint. We find that the **linear probe accuracy continues to increase even after 1500 epochs of training** particularly for larger models, justifying the need to pretrain for a large number of epochs (see fig. A.21).

Centered kernel alignment or CKA (Kornblith et al., 2019) measures the similarity between representations of two different networks. We use CKA to measure the similarity between the representations of MAEs trained with different number of encoder layers and with 4 decoder layers. White indicates that the similarity is high and black/red indicates that the similarity is low. We find that the larger networks are more to the 12-layer encoder while while the smaller networks are less similar to the 12-layer network, particularly at the last layer.

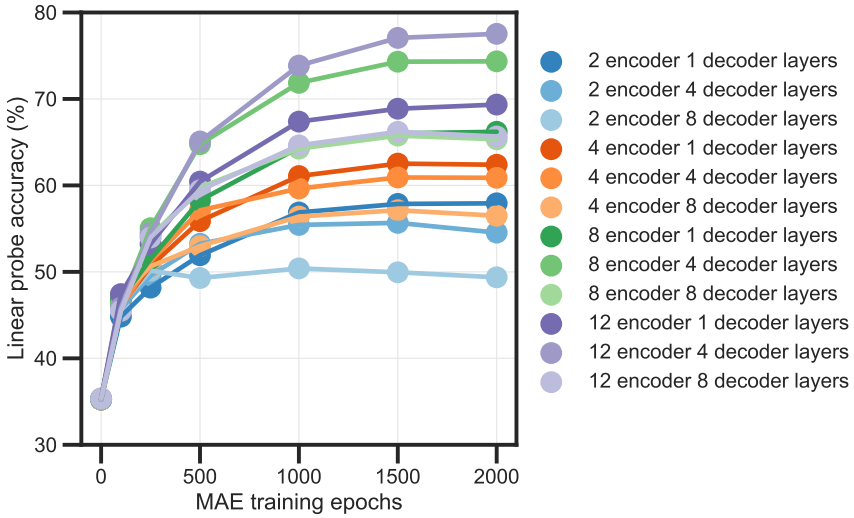


Figure A.24: We plot the linear probe accuracies of different masked autoencoders over the course of training. They accuracy continue to increase even after 1000 epochs of training, justifying the need for long training times. Larger models tend to require longer training times.

A.5.1 More experiments with the Ising model

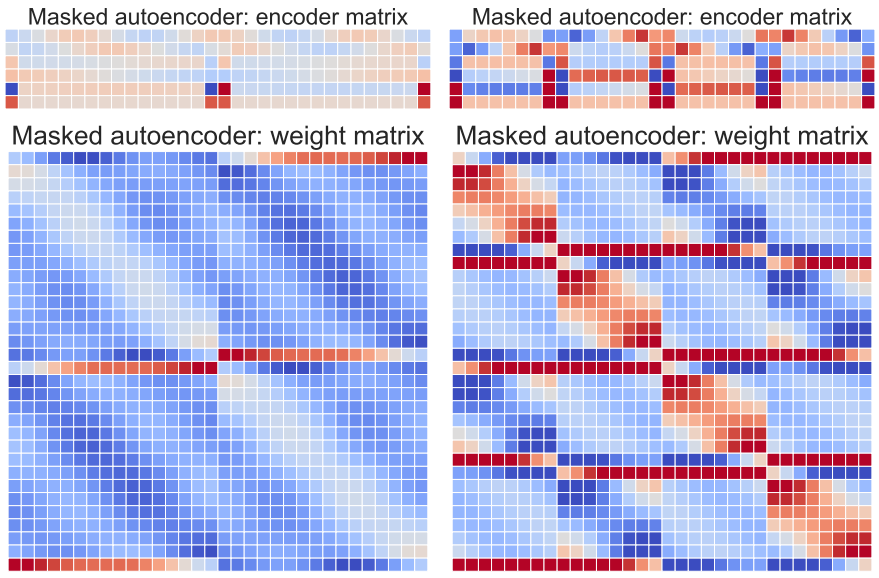


Figure A.26: We plot the weight matrix AB and the encoder matrix A for **(left)** patch-size 16 and masking ratio of 0.5 and **(right)** patch-size 8 and masking ratio 0.5. Increasing the patch-size while keeping the masking ratio fixed biases the encoder towards features that capture long-range correlations.

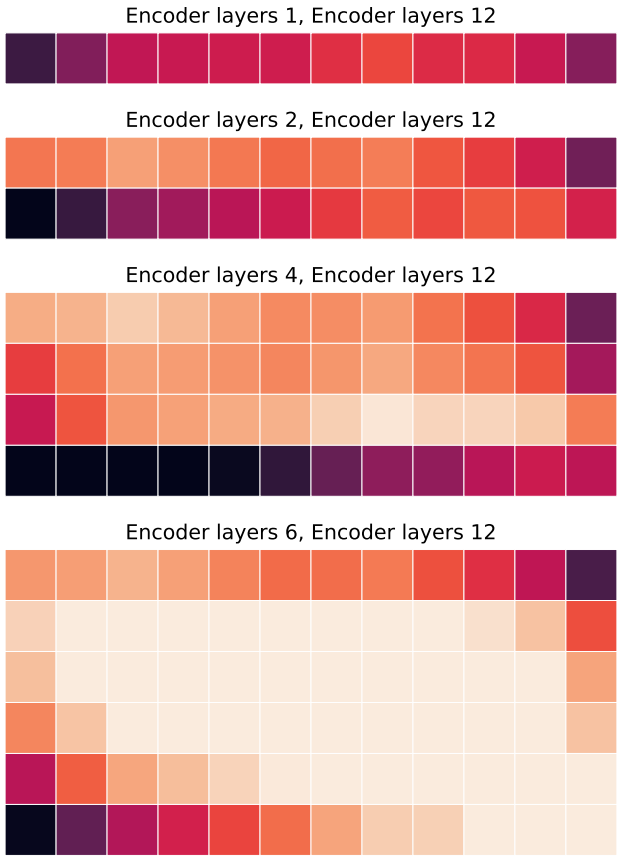


Figure A.25: CKA between MAEs trained with different number of encoder layers. Each row and column corresponds to the similarity between a k-layer encoder and the 12-layer encoder (hence 12 columns). The darker shades indicate that the representations for those two layers are not similar.

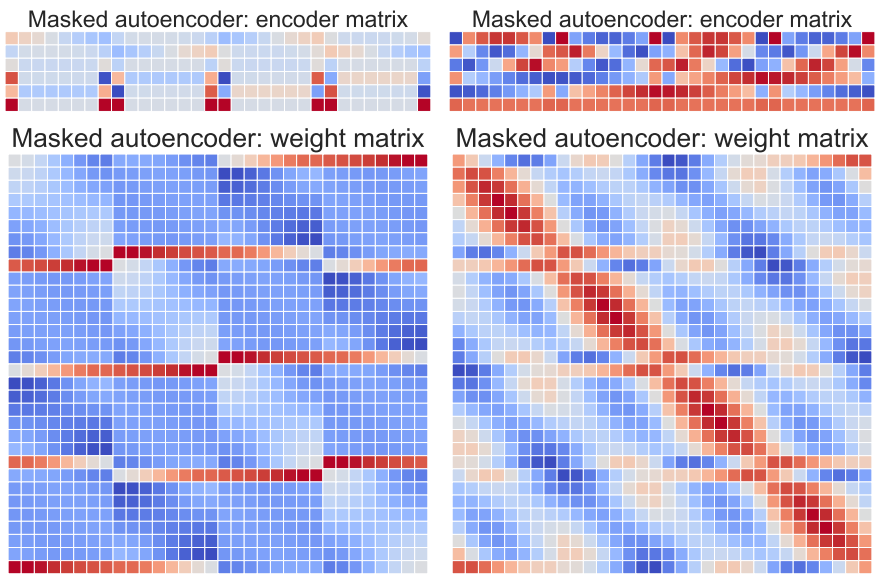


Figure A.27: We plot the weight matrix AB and the encoder matrix A for **(left)** patch-size 16 and masking ratio of 0.99 and **(right)** patch-size 8 and masking ratio (0.01). Reducing the masking ratio biases the encoder towards features based on local correlations while increasing the masking ratio prioritizes features that capture long range correlations.

A.6 Experiment details

A.6.1 Experimental details for Model Zoo

For all 3 networks, the final pooling layer is replaced with an adaptive pooling layer in order to handle input images of different sizes. Convolutional layers are initialized using the Kaiming-Normal initialization. The bias parameter in batch normalization is set to zero with the affine scaling term set to one. The bias of the final classification layer is also set to zero; this helps keep the logits of the different tasks on a similar scale.

Optimization All models are trained in mixed-precision (32-bit weights, 16-bit gradients) using Stochastic Gradient Descent (SGD) with Nesterov's acceleration with momentum coefficient set to 0.9 and cosine annealing of the learning rate schedule for 200 epochs. Training of any model with multiple tasks involves mini-batches that contain samples from all tasks.

Hyper-parameter optimization We used Ray Tune (Liaw et al., 2018) for hyper-parameter optimization. The Async Successive Halving Algorithm (ASHA) scheduler (Li et al., 2020b) was used to prune hyper-parameter choices with the search space determined by Nevergrad (Rapin and Teytaud, 2018). The mini-batch size was varied over [8, 16, 32, 64]; the logarithm (base 10) of the learning rate was sampled from a

uniform distribution on $[-4, -2]$; dropout probability was sampled from a uniform distribution on $[0.1, 0.5]$; logarithm of the weight decay coefficient was sampled from $[-6, -2]$. We used a set of experiments for continual learning on the Coarse-CIFAR100 dataset with different samples/class (100 and 500) to perform hyper-parameter tuning.

The final values of training hyper-parameters that were chosen are, learning-rate of 0.01, mini-batch size of 16, dropout probability of 0.2 and weight-decay of 10^{-5}. Model Zoo uses $\ell = \min(k, 5)$ at each round of continual learning where n is the number of tasks; for tasks with only 5 tasks (MNIST-variants) we use $\ell = 2$. We did not tune these two hyper-parameters using Ray because it is quite cumbersome to do so. We selected these values manually across a few experiments; changing them may result in improved accuracy for Model Zoo.

The final values of training hyper-parameters. The chosen values are, learning-rate of 0.01, mini-batch size of 16, dropout probability of 0.2 and weight-decay of 10^{-5}.

Model Zoo uses $\ell = \min(k, 5)$ at each round of continual learning where n is the number of tasks; for tasks with only 5 tasks (MNIST-variants) we use $\ell = 2$. We did not tune these two hyper-parameters using Ray because it is quite cumbersome to do so. We selected these values manually across a few experiments; changing them may result in improved accuracy for Model Zoo.

All hyper-parameters are kept fixed for all datasets, architectures, and experimental settings . We are interested in characterizing the performance of Model Zoo and its variants across a broad spectrum of problems and datasets. While we believe we can get even better numerical accuracy, by tuning hyper-parameters specially for each problem, we do not so for the sake of simplicity. As the main paper discusses, we outperform existing methods quite convincingly across the board in both multi-task and continual learning.

Data augmentation MNIST and CIFAR10/100 datasets use padding (4 pixels) with random cropping to an image of size 28×28 or 32×32 respectively for data augmentation. CIFAR10/100 images additionally have random left/right flips for data augmentation. Images are finally normalized to have mean 0.5 and standard deviation 0.25. Split-miniImagenet uses the same augmentation as CIFAR-10 and CIFAR-100. We use augmentations even in the single epoch setting, although it is not beneficial to do so.

A.6.2 Experimental Details for non-monotonic trends in data

We consider two types of setups to study the impact of OOD data:

OOD data arising due to geometric intra-class nuisances We study the effect of intra-class nuisances using a classification task as the target task and transformed versions of the same task as different OOD tasks. In this regard, we consider the following experimental setups.

1. **Rotated MNIST: unrotated-domain task as target and θ°- rotated-domain task as OOD:** We consider the 10-way classification of unrotated images as the target task and that of the θ°- rotated images as the OOD samples. We can have different OOD tasks by selecting different values for θ.

2. **Rotated CIFAR-10: T_2 as target and rotated T_2 as OOD:** We choose the bird vs. cat (T_2) task from Split-CIFAR10 as the target task. We then rotate the images of T_2 by an angle θ° counter-clockwise around their centers to form a new task denoted by θ-T_2, which we consider as the OOD task. Different OOD tasks can be obtained by selecting different values for θ.

3. **Blurred CIFAR-10: T_4 as target and blurred T_4 as OOD:** We choose the Frog vs. Horse (T_4) task from Split-CIFAR10 as the target task. We then add Gaussian blur with standard deviation σ to the images of T_4 to form a new task denoted by σ-T_2, which we consider as the OOD task. By setting distinct values for σ, we can have different OOD tasks.

OOD data arising due to category shifts and concept drifts We study this aspect using two different target and OOD classification problems as described below.

1. **Split-CIFAR10: T_i as Target and T_j as OOD:** We choose a pair of distinct tasks from the 5 binary classification tasks of Split-CIFAR10 and consider one as the target task and the other as the OOD task. We perform experiments for all pairs of tasks (20 in total) in Split-CIFAR10.

2. **PACS (Li et al., 2017): Photo-domain task as target and X-domain task as OOD:** Out of the four 3-way classification tasks from PACS, we select the photo-domain task as the target task and consider one of the remaining 3 domain tasks (for instance, the sketch-domain task) as the OOD task.

3. **DomainNet (Peng et al., 2019): Real-domain task as target and X-domain task as OOD:** Out of the six binary classification tasks from DomainNet we consider the real-domain task as the target task and select one of the remaining 5 domain tasks (for instance, the painting-domain task) as the OOD task.

4. **CINIC-10: CIFAR task as target and ImageNet task as OOD:** Here we simply select the 10-way classification of CIFAR images as the target task and that of ImageNet as the OOD task.

A.6.3 Experimental details for prospective learning

A.6.3.1 Training and evaluation

Training setup. Each learner receives a t-length sequence of samples $z_{\leq t}$ drawn from the stochastic process, as the training data. Upon training, the learner is expected to make predictions on future samples that correspond to times $t' > t$ up to a fixed horizon T. At each future time t', we do not train (modify the weights) using samples after time t (because we do not have them, but we will make predictions on these samples). Given samples $z_{\leq t}$, a time-aware hypothesis class minimizes the empirical prospective risk

$$\hat{\ell}_t(h, Z) = \frac{1}{t} \sum_{s=1}^{t} \ell(s, h_s(x_s), y_s);$$

For an MLP or CNN, h_s corresponds to a network that takes time s as input.

Hyper-parameters All the networks are trained using stochastic gradient descent (SGD) with Nesterov's momentum and cosine-annealed learning rate. The networks are trained at a learning rate of 0.1 for the synthetic tasks, and learning rate of 0.01 for MNIST and CIFAR. The weight-decay is set to 1×10^{-5}. The images from MNIST and CIFAR-10 are normalized to have mean 0.5 and standard deviation 0.25. The models were trained for 100 epochs, which is many epochs after achieving a training accuracy of 1.

Evaluation We estimate the prospective risk of each learner using a Monte Carlo estimate. For a given training dataset $z_{\leq t}$, we estimate a sequence predictors $h \equiv (h_t)$ which we use to make predictions on future samples. We wish to approximate the prospective risk (Equation (2.9)) for the estimated sequence of

predictors. We do so, for a single future realization $z_{>t}$ of this process, which yields the estimate

$$\hat{R}_t(h) = \frac{1}{(T-t)} \sum_{s=t+1}^{T} \ell\left(s, h_s(x_s^j), y_s^j\right).$$

In our experiments, $T = 50{,}000$ for CIFAR-10 and MNIST while $T = 10{,}000$ for the synthetic data experiments. For a single learning algorithm, we compute the empirical prospective risk at 15-40 different time steps which results in a significant number of GPU hours in order to plot the learning curves. For every time step, we compute the mean and standard deviation of the empirical prospective risk using 5 random seeds.

A.6.3.2 Architectural Details

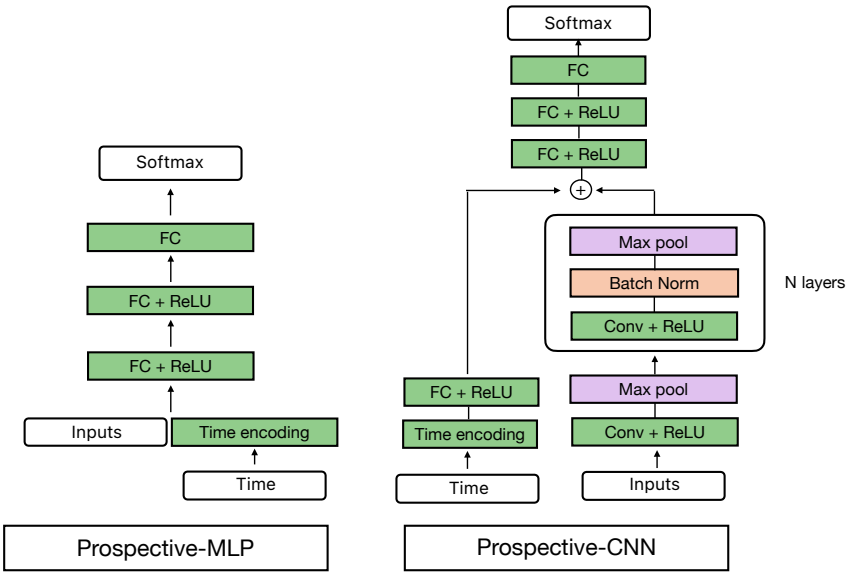


Figure A.28: Schematic illustration prospective-MLP and prospective-CNN.

We considered the following architecture choices for the time-agnostic restropective algorithms like ERM that ignore time and the ordering associated with the samples in $z_{\leq t}$.

Retrospective-MLP/CNN. A multi-layer perceptron (MLP) with two hidden layers with 256 units is used for the synthetic tasks and the MNIST task. For CIFAR-10, we use a a small convolutional network with 0.12M parameters. It comprises of 3 convolution layers (kernel size 3 and 80 filters) interleaved with max-pooling, ReLU, batch-norm layers, with a fully-connected classifier layer.

Prospective ERM with MLP and CNNs. In order to incorporate time into the hypothesis class, we consider an embedding function $\varphi : \mathbb{R} \to \mathbb{R}^d$ that takes raw time as an input and returns a d-dimensional vector denoted as the time-embedding. In our experiments, we define $\varphi : \mathbb{R} \to \mathbb{R}^d$ as a function that maps

$$t \mapsto (\sin(\omega_1 t), \ldots, \sin(\omega_{d/2} t), \cos(\omega_1 t), \ldots, \cos(\omega_{d/2} t)),$$

where, $\omega_i = \pi/i,\ i = 1, \ldots, d/2$ to the be the collection of angular frequencies. We briefly discuss the rationale for this choice in Figure A.30. In our experiments, we use $d = 50$.

We make our classifiers a function of time by including time t as an input the neural network. This allows the network to vary its hypothesis over time. For MLPs, we concatenate the input with its corresponding time-embedding $\varphi(t)$ which is fed as input. For the CNN (see Figure A.28), we add the time-embedding to the output of the convolutional layers instead of concatenating it to the inputs. We also tried concatenating the time-encoding to the inputs of the CNN but found that it performed poorly in both scenarios 2 and 3 (see Figure A.29).

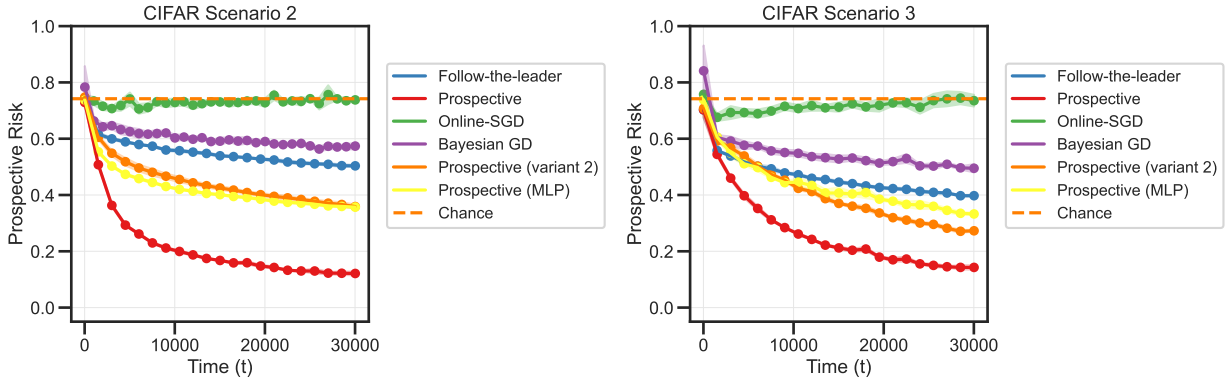


Figure A.29: Prospective risk of the CNN architecture on CIFAR-10 for scenarios 2 and 3. The performance of the CNN architecture is significantly worse when the time-embedding is concatenated to the input (variant 2).

Frequencies for embedding time In the original Transformer architecture, Vaswani et al. (2017) use a position-embedding using the frequencies $\omega_i = 1/10000^{2i/d}\ \ i = 1, \ldots, d/2$. There are two key differences: (1) We use the absolute time t instead of the relative position, (2) We use the angular frequencies $2\pi/i$. In Figure A.30 *(right)*, we illustrate the time-embeddings when we use the two different choices for angular frequencies. For $d = 128$, we find that the frequencies from Vaswani et al. (2017) result in slowly changing

features which makes it less suitable for our task, i.e., many of the dimensions are constant over time which makes many of the dimensions uniformative for the task. In our experiments, we found out that MLPs and CNNs that use the frequencies from Vaswani et al. (2017) perform poorly on the MNIST task for scenarios 2 and 3.

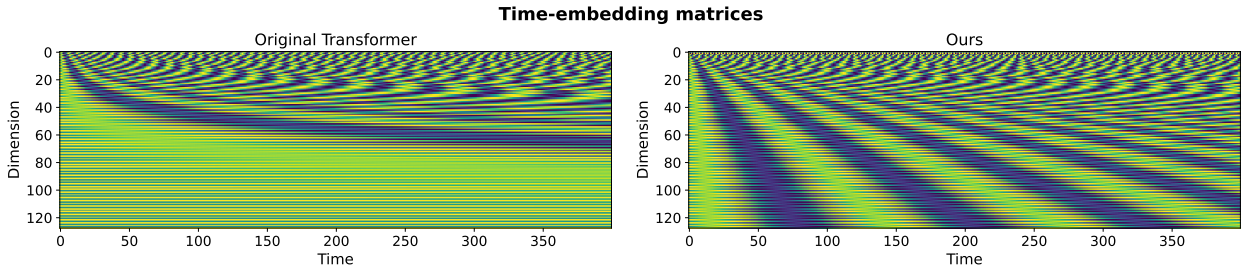


Figure A.30: The time-embeddings computed using (1) frequencies from Vaswani et al. (2017) *(left)*, and (2) the frequencies from proposed in our work *(right)*.

A.6.3.3 Prompts for testing prospective learning in LLMs

We use the following 3 prompts to generate a sequence of predictions using in LLama-7B and Gemma-7B. We found that the LLMs always generated a sequence of 0s and 1s and we did not need to post-process the response or change how the tokens were sampled. We generate 20 samples using greedy decoding; the language models are executed with the weights in 16-bit precision. We tried a few different variants for providing prompts to the LLM, e.g., by adding spaces between the 1s and 0s, the results are qualitatively similar.

Scenario 1

Consider the following sequence of outcomes generated from a single Bernoulli distribution.

11110101111110111111111111

The next 20 most likely outcomes are:

Scenario 2

Consider the following sequence of outcomes generated by two Bernoulli distributions, where all even outcomes are generated by a Bernoulli distribution with parameter 'p' and odd outcomes are generated from a Bernoulli distribution with parameter '1-p'.

10101010101010101010101000101010101010101

The next 20 most likely sequence of outcomes are:

Scenario 3

Consider the following sequence of states generated by a Markov process with 2 states (0, 1):

10101101010100101010

The next 20 most likely outcomes are:

To make the LLM generate a sequence of Bernoulli trials with probability 0.75, we used the following prompt.

Bernoulli trials

Generate outcomes of 10 Bernoulli trials where 0 is generated with probability 0.25 and 1 with probability 0.75

A.6.4 Experimental details for showing that trajectories of representations are low-dimensional

Data. We performed experiments using two datasets.

1. CIFAR10 has 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck) with RGB images of size 32×32, and
2. ImageNet has 1000 classes each with about 1000 RGB images of size 224×224.

ImageNet classes are derived from the leaves of the Wordnet hierarchy (Miller, 1995). We use this hierarchy to create tasks using different subsets of ImageNet; We use all classes under a node to create a task. The tasks that we consider are: Dogs, Vertebrates, Invertebrates, Instrumentality, Reptile and Birds. We also consider a task with 333 randomly selected classes and unlike other tasks, it spans many different phyla of ImageNet.

Architectures. We use a Wide-Resnet (Zagoruyko and Komodakis, 2016) architecture for experiments on supervised learning on CIFAR-10 (depth 16 and widening factor 4) and a Resnet-18 (He et al., 2016) to train a model using SimCLR. All experiments on ImageNet use the Resnet-50 architecture.

All convolutional layers are initialized using the Kaiming-Normal initialization. For the Wide-Resnet, the final pooling layer is replaced with an adaptive pooling layer in order to handle input images of different sizes.

We make three modifications to these architectures.

1. We remove the bias from the final classification layer; this helps keep the logits of the different tasks on a similar scale.

2. In the experiments for Result 3 (episodic meta-learning) and Result 6 (fine-tuning), we replace batch normalization with layer norm in the Wide-Resnet. This is because we found in preliminary experiments that batch-normalization parameters make training meta-learning models very sensitive to choices of hyper-parameters (e.g., the support or query shot), and that the learned representations of new tasks were quite different in terms of their predictions (and thereby the Bhattacharyya distance) but all the difference was coming from modifications to the BN parameters.

3. In the Resnet-50, we replace the pooling layers with BlurPool (Zhang, 2019). The bias parameter in batch normalization is set to zero with the affine scaling term set to one.

Training procedure All models are trained in mixed-precision (32-bit weights, 16-bit gradients) using stochastic gradient descent (SGD) with Nesterov's acceleration with momentum coefficient set to 0.9 and cosine annealing of the learning rate schedule. Batch-normalization parameters are excluded from weight decay.

CIFAR10 datasets use padding (4 pixels) with random cropping to an image of size 28×28 or 32×32 respectively for data augmentation. CIFAR10 images additionally have random left/right flips for data augmentation. Images are finally normalized to have mean 0.5 and standard deviation 0.25.

Supervised learning models (including fine-tuning) for CIFAR10 are trained for 100 epochs with a batch-size of 64 and weight decay of 10^{-5} using the Wide-Resnet.

Episodic meta-learners are trained using a Wide-Resnet and with the prototypical loss (Snell et al., 2017). For the 2-way meta-learner, each episode contains 20 query samples and 10 support samples. For the 5-way meta-learner, each episode contains 50 query samples and 10 support samples. We found (Result 4) to hold across different choices of these hyper-parameters in small-scale experiments. Models are trained for around 750 epochs and the episodic learner is about 5 times slower to train with respect to wall-clock time.

We train models using SimCLR on CIFAR10 and on tasks created from CIFAR10. For the augmentations, we use random horizontal flips, random grayscale, random resized crop and color jitter. Models are trained for 200 epochs for 2-way classification problems and for 500 epochs when trained on the entirety of CIFAR10 with the Adam optimizer and an initial learning rate of 0.001.

A.6.4.1 Experiments on ImageNet

We make use of FFCV (Leclerc et al., 2023). which is a data-loading library that replaces the pytorch Dataloader. FFCV reduces the training time on ImageNet to a few hours, which allows us to train 100s of models on ImageNet, or on tasks created from it. Our implementation of ImageNet training builds on the FFCV repository. ImageNet models are trained for 40 epochs with progressive resizing – the image size is increased from 160 to 224 between the epochs 29 and 34. Models are trained on 4 GPUs with a batch-size of 512. The training uses two types of augmentations – random-resized crop and random horizontal flips. Additionally, we use label smoothing with the smoothing parameter set to 0.1.

A.6.4.2 Implementing InPCA in very high dimensions

We calculate an InPCA embedding of models along multiple trajectories, e.g., a typical experiment has about 25 trajectories (multiple random seeds, tasks, or representation learning methods) and about 50 models (checkpoints) along each trajectory. Each model is a very high-dimensional object (with dimensionality NC where $N \sim 10^5$ and $C \sim$ 10-10^3). Even if the matrix D in eq. (4.9) is relatively manageable with $n \sim 1250$, each entry of D is $\mathrm{d_B}(P_u, P_v)$ and therefore requires $\sim 10^8$ operations to compute. Implementing InPCA—or even PCA—for such large matrices requires a large amount of RAM. We reduced the severity of this issue

to an extent using Numpy's memmap functionality. Also note that calculating only the top few eigenvectors of eq. (4.9) suffices to visualize the models, we do not need to calculate all.

The formula eq. (4.2) is an effective summary of the discrepancies between how the predictions made by two probabilistic models differ; even small differences in two models, e.g., even if both P_u and P_v make mistakes on exactly the same input samples, if $p_u^n(c)$ is slightly different than $p_v^n(c)$ for even one of n or c, the divergence is non-zero. InPCA is capable of capturing the differences between two such models eq. (4.9). However, when the number of classes is extremely large, the number of terms in the summation is prohibitively large and analyzing the discrepancies or calculating the embedding becomes rather difficult.

We also developed a method to work around this issue. We can use a random stochastic matrix (whose columns sum up to 1) to project the outputs for each sample $\{p_u^n(c)\}_{c=1,\ldots,C}$ into a smaller space before calculating eq. (4.2). This amounts to pretending as if the model predicts not the actual classes but a random linear combination of the classes (even if the model is trained on the actual classes). This is a practical trick that is necessary only when we are embedding a very large number of very high-dimensional probabilistic models. We checked in our Imagenet experiments that using this trick gives the same embeddings.

In this paper, we did not need to use this projection trick. However, we found that this tricks makes it computationally faster to compute the embeddings and we have seen it to work well in practice. We have shared the code for this procedure, since it allows other people to reproduce the results using fewer computational resources.

A.6.5 Experimental details for showing redundancy in perception tasks

A.6.5.1 Datasets

We examine various complex perception tasks, including classification, semantic segmentation, optical flow, and depth/disparity prediction. In this work, we use CIFAR-10 for classification, ImageNet for classification, the Cityscapes dataset for semantic segmentation and disparity prediction(Cordts et al., 2016), the ADE20K for semantic segmentation(Zhou et al., 2017) and an augmented version of the M3ED dataset(Chaney et al., 2023). We feature complex real scenes beyond just classification datasets. For example, the M3ED dataset has many natural scenes, including cars driving in urban, forest, and daytime and nighttime conditions.

A.6.5.2 Projecting data onto different subspaces

A key operation in this work is projecting input data onto different subspaces using PCA, Fourier and wavelet bases.

Indices for the bases. For each basis, we define an index that determines the ordering of the basis elements. The indices are defined as follows:

(1) For PCA, we order the basis elements by the descending order of the eigenvalues. The basis element with index 5 is the eigenvector corresponding to the 5th largest eigenvalue.

(2) For the Fourier basis, the basis elements are ordered by the increasing radial frequency. The basis elements with index i, contains all spatial frequencies (ω_1, ω_2) such that

$$i \leq \sqrt{\omega_1^2 + \omega_2^2} < i + 1.$$

(3) For wavelets, the basis elements are ordered by increasing scales. The small scales capture low-frequency information, while the larger scales capture high frequency information. For example, a 2-level wavelet decomposition of an image contains 7 coefficients. The approximation coefficient (cA) represents the smallest scale. Each level has 3 detail coefficients (cH, cV, cD) representing the horizontal, vertical, and diagonal details, respectively. The ordering of the wavelet coefficients by scale(small-large) is (cA2, cH2, cV2, cD2, cH1, cV1, cD1). Hence the basis elements with index 5 correspond to the coefficients cH1.

Bands. We use the ordering of the basis elements to define different "bands" of the basis. Low pass and high pass filters consider basis elements that are lesser than or greater than a certain index. Band pass filters consider basis elements between two indices. We train networks after projecting the input data onto different subspaces defined by the low pass, high pass, and band pass filters. After projecting onto these subspaces, we transform the input back to pixel space and train the network.

We perform experiments with PCA, Fourier and Wavelet bases for CIFAR-10. However, computing PCA on datasets as large as ImageNet is computationally difficult, so we study the larger datasets using the Fourier or wavelet bases. In fig. 4.9, we find a strong correspondence between the two, which suggests that one should

get similar results using the PCA, Fourier and wavelet bases.

A.6.5.3 Neural network architectures and training procedure

Dataset	e_{tot}	λ	λ_{max}	**Batch Size**	**Channels**			**# Samples**	
					In	**Out**	**Base**	**Train**	**Test**
CS Depth	50	0.0001	0.005	128	6	1	64	22972	500
M3ED Depth	40	0.001	0.01	128	6	1	64	34255	15020
M3ED Flow	40	0.001	0.01	128	6	2	32	34255	15020
M3ED Sem	60	0.005	0.0001	128	3	20	64	54793	13822
Cityscapes Sem	40	0.0005	0.01	128	3	20	64	22973	500
ADE20k Sem	60	0.0001	0.01	128	3	150	64	20210	2000

Table A.4: Hyperparameters and dataset sizes for different datasets used to test redundancy

In our experiments, we apply various low-pass/high-pass/band-pass filters and train networks based on the results of these filters. We then measure the test error (1-Accuracy, for classification and segmentation), the mean average error (for depth estimation), or the average endpoint error (optical flow). The different filters can tell us which subspaces of the input have information relevant for classification–for example, if a high-pass filter results in chance accuracy on a classification task, then this suggests that there is little information for the task present in the tail of the spectrum. However, our results suggest otherwise, the entire spectrum performs well on our tasks.

Classification. For CIFAR-10, we use a Wide-Resnet (Zagoruyko and Komodakis, 2016) with 16 layers and 4 blocks of sizes 16, 64, 128 and 256. We normalize the images by the mean and standard deviation but do not apply any augmentations during training. The models are trained for 100 epochs using stochastic gradient descent with Nesterov momentum (of 0.9) and use a learning rate of 0.05 with a batch size of 64 and weight decay of 5×10^{-5}. The networks are trained to optimize the cross-entropy loss.

The ImageNet models are trained using the Resnet-50 (He et al., 2016) architecture with the pooling layers replaced with BlurPool (Zhang, 2019). To speedup training, we use FFCV (Leclerc et al., 2023), which is a data-loading library optimized to load and perform augmentations on the training data quickly. We train the models for 40 epochs with progressive resizing – the image size is increased from 160 to 224 between

the epochs 29 and 34. The projection operation (band pass, low pass, or high pass) is applied to an image of size 256 regardless of the stage of training in progressive resizing. Models are trained on 4 GPUs with a batch-size of 512. We use random-resized crop and random horizontal flips as the two augmentations and label smoothing parameter set to 0.1. We use stochastic gradient descent as the optimizer with the learning rate schedule that is annealed linearly. We use a weight decay of 0.0001 but do not apply it to the batch norm parameters.

Semantic segmentation, Optical Flow and Depth prediction. All images down-sampled to 200×200, except for M3ED semantic segmentation which is 180×200. We apply filters starting from 10- half the minimum resolution for the various low-pass and high-pass filters. As for band-pass filters, we again apply them using the same interval with a width of 10 for all dense perception experiments. For each experiment, one of these filters is applied to the input image spectra and used as input for the model of our task. We tested most of our models on a U-Net (Ronneberger et al., 2015) architecture with 3 layers with a 2x down-sampling and channel multiplier. These backbones contain downsampling/upsampling operations; therefore, to ensure a valid sized output, we pad our input evenly and crop the output for loss calculation. We train all models using supervised training with backpropagation using AdamW (Loshchilov and Hutter, 2017) with OneCycleLR (Smith and Topin, 2017) policy with an initial learning rate of λ and a maximum learning rate of λ_{max} with a batch size of 128 for e_{tot} epochs and fp16/bf16 weights.

Now, we will review the respective input and training loss used for each task. The task of optical flow takes in two sequential in time RGB images($6 \times M \times N$) and regresses optical flow($2 \times M \times N$) with 2 channels corresponding to the flow in the x and y directions. The loss function for optical flow is the robust (Charbonnier et al., 1994) comparing the ground truth and estimated optical flow along with a second-order smoothness loss weighted with a penalty of $\beta = 0.5$. In both cases, our depth prediction takes in two frames, for M3ED this corresponds to two sequential frames and for Cityscapes this corresponds to a left and right camera(in this case we predict disparity). We again use the robust (Charbonnier et al., 1994) comparing the ground truth and estimated depth. Finally, our semantic segmentation task takes as input one RGB image ($3 \times M \times N$) and regresses a distribution of semantic classes($K \times M \times N$), for classes K. The loss function for semantic segmentation was the cross entropy loss between the predicted class distribution and actual class

distribution per pixel. These tasks are masked based on whether ground truth labels are available for this output pixel.

A.6.5.4 Random bands used in experiments

	Center Frequency	Width
Depth	5, 30, 50, 80	2
	10, 50, 80	5
	5, 45, 75	5
	5, 40, 60, 80	2
	15, 45, 85	5
Semantic Segmentation	5, 30, 50, 80	2
	10, 50, 80	5
	5, 45, 75	5
	5, 40, 60, 80	2
	15, 45, 85	5
Optical Flow	25, 40, 60, 80	[16, 8, 4, 2]
	5, 40, 60, 80	2
	30, 55, 70, 90	2
	5, 10, 30, 50, 70, 90	2
	30, 60, 80, 93	[16, 8, 4, 2]
	17, 50, 70, 90	[16, 8, 4, 2]
	10, 30, 50, 70, 90	6
	5, 30, 50, 80	2
	17, 50, 70, 90	5

Table A.5: Random bands used in fig. 4.12. For each of the experiments, we list the center frequency used and the corresponding width of that pass band. For experiments with multiple widths, each width is used for the corresponding sequentially ordered center frequency.

A.6.5.5 Experimental setup for audition tasks

We used a cochlear model with two stages of filters: 42 gammatone spectral filters followed by a temporal filter (Zhang et al., 2001; Lewicki, 2002; Tabibi et al., 2017) to approximate the structure of audio inputs in our task to that of the auditory system. These gammatone filters had center frequencies between 22.9 Hz to 20208 Hz, which covered the frequency range of all vocalizations studied. The temporal filter was implemented as a difference of two kernels of the form:

$$g(n) = an^m e^{-bn},$$

in which n is in units of samples and a, b, and m are filter parameters. The temporal filter was created by taking the difference between g_1 and g_2 with the following parameters: g_1: $a = 1.5$, $b = 0.04$, and $m = 2$; and g_2: $a = 1$, $b = 0.036$, and $m = 2$. This parameter set accounts for some key aspects of cochlear temporal processing (Lyon et al., 2010; Tabibi et al., 2017). Each filter output was normalized to have zero mean and unit standard deviation.

The conceptual underpinning of Slow Feature Analysis (SFA) is the "slowness principle". This principle hypothesizes that higher-order information in a stimulus (e.g., stimulus identity) changes at a slower time scale than other lower-order fluctuations (e.g., acoustic features). SFA learns the slow features in a stimulus through unsupervised learning. The formulation for linear SFA is described in the following equations:

$$\mathbb{R}^m \ni y(t) = \hat{w}^\top x(t)$$
$$\hat{w} = \underset{w \in \mathbb{R}^{d \times m}}{\operatorname{argmin}} \left\langle \left\| y(t) - y(t-1) \right\|_2^2 \right\rangle;$$

where $x(t) \in \mathbb{R}^d$ is the auditory stimulus (output of the cochlear model), $w \in \mathbb{R}^{d \times m}$ is the set of slow features and $y(t)$ is the projection of the stimulus into the feature space. The notation $\langle\rangle$ denotes an expectation over time. Slowness is realized by finding m features that minimize the average squared temporal difference of $y(t)$. This effectively assigns weights to the original stimulus features to generate an output signal $y(t)$ that changes most slowly across time. To obtain non-trivial solutions, one imposes the constraint that the SFA features $y(t)$ have zero-mean and unit variance across time, and are uncorrelated with each other. Features found by SFA are the eigenvectors of the covariance matrix of the derivative of the auditory stimulus $x(t)$, arranged from the smallest eigenvalue to the largest, i.e., slowest to fastest.

We use a multi-layer perceptron (MLP) to discriminate between vocalization pairs. The input layer has dimensionality equal to the number of SFA features, the hidden layer with ReLU nonlinearities has 5 neurons and a single output neuron to discriminate between two categories. Adam is used to fit this MLP with a 50-50 stratified split of the train and test data; we used bootstrap (5 runs) to report test errors.

APPENDIX B

BACKGROUND

B.1 Fisher's linear discriminant

B.1.1 OOD-Agnostic Fisher's Linear Discriminant

In this section, we derive FLD when we have samples from a single task – which is also applicable to the OOD-agnostic (when the identity of the OOD samples are not known) setting. Consider a binary classification problem with $D_t = \{(x_i, y_i)\}_{i=1}^n \sim P_t$ where $x_i \in X \subseteq \mathbb{R}^d$ and $y_i \in Y = \{0, 1\}$.

Let f_k and π_k be the conditional density and prior probability of class k ($k \in \{0, 1\}$) respectively. The probability that x belongs to class k is

$$p(y = k \mid x) = \frac{\pi_k f_k(x)}{\pi_0 f_0(x) + \pi_1 f_1(x)},$$

and the maximum *a posteriori* estimate of the class label is

$$h(x) = \underset{k \in \{0,1\}}{\operatorname{argmax}}\ p(y = k \mid x) = \underset{k \in \{0,1\}}{\operatorname{argmax}}\ \log\big(\pi_k f_k(x)\big). \tag{B.1}$$

Fisher's linear discriminant (FLD) assumes that each f_k is a multivariate Gaussian distribution with the same covariance matrix Σ, i.e,

$$f_k(x) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_k)^\top \Sigma^{-1}(x - \mu_k)\right).$$

Under this assumption, the joint-density f of (x, y) becomes,

$$f(x, y) \propto \prod_{k=0}^{1} \left[\frac{\pi_k}{|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_k)^\top \Sigma^{-1}(x - \mu_k)\right)\right]^{\mathbf{1}[y=k]}$$

Therefore, the log-likelihood $l(\mu_0, \mu_1, \Sigma, \pi_0, \pi_1)$ over D_t is given by,

$$\ell(\mu_0, \mu_1, \Sigma, \pi_0, \pi_1) = \sum_{k=0}^{1} \sum_{(x,y)\in D_{t,k}} \left[\log \pi_k - \frac{1}{2} \log |\Sigma| - \frac{1}{2}(x - \mu_k)^\top \Sigma^{-1}(x - \mu_k) \right] + \text{const.}$$

where $D_{t,k}$ is the set of samples of D_t that belongs to class k. Based on the likelihood function above, we can obtain the maximum likelihood estimates $\hat{\mu}_k, \hat{\Sigma}, \hat{\pi}_k$. The expression for the estimate $\hat{\mu}_k$ is

$$\hat{\mu}_k = \frac{1}{|D_{t,k}|} \sum_{(x,y)\in D_{t,k}} x. \tag{B.2}$$

Plugging these estimates into eq. (B.1), we get,

$$\begin{aligned}
\hat{h}(x) &= \underset{k\in\{0,1\}}{\text{argmax}} \left[\log \hat{\pi}_k - \frac{1}{2} \log |\hat{\Sigma}| - \frac{1}{2}(x - \hat{\mu}_k)^\top \hat{\Sigma}^{-1}(x - \hat{\mu}_k) \right] \\
&= \underset{k\in\{0,1\}}{\text{argmax}} \left[\log \hat{\pi}_k - \frac{1}{2} \log |\hat{\Sigma}| + x^\top \hat{\Sigma}^{-1} \hat{\mu}_k - \frac{1}{2} \hat{\mu}_k^\top \hat{\Sigma}^{-1} \mu_k \right]
\end{aligned}$$

Therefore, $\hat{h}(x) = 1$ iff,

$$\begin{aligned}
x^\top \hat{\Sigma}^{-1} \hat{\mu}_1 - \frac{1}{2} \hat{\mu}_1^\top \hat{\Sigma}^{-1} \mu_1 + \log \hat{\pi}_1 &> x^\top \hat{\Sigma}^{-1} \hat{\mu}_0 - \frac{1}{2} \hat{\mu}_0^\top \hat{\Sigma}^{-1} \mu_0 + \log \hat{\pi}_0 \\
x^\top \hat{\Sigma}^{-1} \hat{\mu}_1 - x^\top \hat{\Sigma}^{-1} \hat{\mu}_0 &> \frac{1}{2} \hat{\mu}_1^\top \hat{\Sigma}^{-1} \mu_1 - \frac{1}{2} \hat{\mu}_0^\top \hat{\Sigma}^{-1} \mu_0 + \log \hat{\pi}_0 - \log \hat{\pi}_1 \\
(\hat{\Sigma}^{-1}(\hat{\mu}_1 - \hat{\mu}_0))^\top x &> (\hat{\Sigma}^{-1}(\hat{\mu}_1 - \hat{\mu}_0))^\top \left(\frac{\hat{\mu}_0 + \hat{\mu}_1}{2} \right) + \log \frac{\hat{\pi}_0}{\hat{\pi}_1}
\end{aligned}$$

Hence the FLD decision rule $\hat{h}(x)$ is

$$\hat{h}(x) = \mathbf{1}\left(\omega^\top x > c\right) \tag{B.3}$$

where $\omega = \hat{\Sigma}^{-1}(\hat{\mu}_1 - \hat{\mu}_0)$ is a projection vector and $c = \omega^\top \left(\frac{\hat{\mu}_0 + \hat{\mu}_1}{2}\right) + \log \frac{\hat{\pi}_0}{\hat{\pi}_1}$ is a threshold. When $d = 1$ and $\pi_0 = \pi_1$, the decision rule reduces to

$$\hat{h}(x) = \mathbf{1}\left(x > \frac{\hat{\mu}_0 + \hat{\mu}_1}{2}\right) \tag{B.4}$$

B.1.2 Deriving the Generalization Error of the Target Task for Synthetic Tasks with FLD

We would like to derive an expression for the average generalization error of the target task, when we consider the synthetic tasks. For simplicity, we set the variance σ^2 of the class conditional densities of the synthetic tasks to 1.

In the OOD-agnostic setting, the learning algorithm sees a single dataset $D = D_t \cup D_o$ of size $n + m$ which is a combination of both target and OOD samples. We can estimate μ_k using eq. (B.2) to obtain

$$\begin{aligned}\hat{\mu}_k = \frac{1}{|D_k|} \sum_{(x,y)\in D_k} x &= \frac{\sum_{(x,y)\in D_{t,k}} x + \sum_{(x,y)\in D_{o,k}} x}{n_k + m_k} \\ &= \frac{n_k \bar{x}_{t,k} + m_k \bar{x}_{o,k}}{n_k + m_k} \\ &= \frac{n\bar{x}_{t,k} + m\bar{x}_{o,k}}{n + m}.\end{aligned} \tag{B.5}$$

where D_k is the set of samples of D that belongs to class k, $n_k = |D_{t,k}|$ and $m_k = |D_{o,k}|$ for $k \in \{0, 1\}$. $\bar{x}_{t,k}$ and $\bar{x}_{o,k}$ denote the sample means of class k in target and OOD datasets respectively. We assume that $\pi = \frac{1}{2}$ from which it follows that $n_k = n\pi_k = \frac{n}{2}$ and $m_k = m\pi_k = \frac{m}{2}$. We cannot explicitly compute $\bar{x}_{t,k}$ and $\bar{x}_{o,k}$ when the OOD samples are not explicitly known, because we cannot separate target samples from OOD samples in D.

Since the samples are drawn from Gaussians, their averages also follow Gaussian distributions. Hence, the threshold $\hat{c} = \frac{\hat{\mu}_0 + \hat{\mu}_1}{2}$ of the hypothesis $\hat{h}$, estimated using FLD, is a random variable with a Gaussian distribution i.e., $\hat{c} \sim \mathcal{N}(\mu_h, \sigma_h^2)$ where

$$\begin{aligned}\mu_h &= \mathbb{E}[\hat{c}] = \frac{m\Delta}{n + m}, \\ \sigma_h^2 &= \mathrm{Var}[\hat{c}] = \frac{1}{n + m}.\end{aligned}$$

The target error of a hypothesis $\hat{h}$ is

$$\begin{aligned} p(\hat{h}(x) \neq y \mid x, \hat{c}) &= \frac{1}{2} p_{x \sim f_{t,1}}[x < \hat{c}] + \frac{1}{2} p_{x \sim f_{t,0}}[x > \hat{c}] \\ &= \frac{1}{2} + \frac{1}{2} p_{x \sim f_{t,1}}[x < \hat{c}] - \frac{1}{2} p_{x \sim f_{t,0}}[x < \hat{c}] \\ &= \frac{1}{2}\big[1 + \Phi(\hat{c} - \mu) - \Phi(\hat{c} + \mu)\big] \end{aligned} \tag{B.6}$$

Using eq. (B.6), the expected error on the target task $e_t(\hat{h}) = \mathbb{E}_{\hat{c} \sim \mathcal{N}(\mu_h, \sigma_h^2)}[p(\hat{h}(x) \neq y \mid x, \hat{c})]$ is given by,

$$\begin{aligned} e_t(\hat{h}) &= \int_{-\infty}^{\infty} \frac{1}{2}\big[1 + \Phi(\hat{c} - \mu) - \Phi(\hat{c} + \mu)\big] \frac{1}{\sigma_h} \phi\left(\frac{\hat{c} - \mu_h}{\sigma_h}\right) d\hat{c} \\ &= \int_{-\infty}^{\infty} \frac{1}{2}\big[1 + \Phi(y\sigma_h + \mu_h - \mu) - \Phi(y\sigma_h + \mu_h + \mu)\big] \phi(y) dy \\ &= \frac{1}{2}\left[\Phi\left(\frac{\mu_h - \mu}{\sqrt{1 + \sigma_h^2}}\right) + \Phi\left(\frac{-\mu_h - \mu}{\sqrt{1 + \sigma_h^2}}\right)\right] \end{aligned}$$

In the last equality, we make use of the identity $\int_{-\infty}^{\infty} \Phi(cx + d)\phi(x)dx = \Phi\big(\frac{d}{\sqrt{1+c^2}}\big)$ where ϕ and Φ are the PDF and CDF of the standard normal. Substituting the expressions for μ_h, σ_h^2 into the above equation, we get

$$e_t(\hat{h}) = \frac{1}{2}\left[\Phi\left(\frac{m\Delta - (n+m)\mu}{\sqrt{(n+m)(n+m+1)}}\right) + \Phi\left(\frac{-m\Delta - (n+m)\mu}{\sqrt{(n+m)(n+m+1)}}\right)\right] \tag{B.7}$$

For synthetic tasks with $\sigma^2 \neq 1$, the target generalization error can be obtained by simply replacing μ and Δ with $\frac{\mu}{\sigma}$ and $\frac{\Delta}{\sigma}$ respectively in eq. (B.7).

B.1.3 OOD-Aware Weighted Fisher's Linear Discriminant

We consider a target dataset $D_t = \{(x_i, y_i)\}_{i=1}^n$ and an OOD dataset $D_o = \{(x_i, y_i)\}_{i=1}^m$, which are samples from the mixture-of-gaussians synthetic task. This setting differs from section B.1.2 since we know whether each sample from $D = D_t \cup D_o$ is OOD or not. This difference allows us to consider a log-likelihood function

that weights the target and OOD samples differently, i.e. we consider

$$\ell(\mu_0, \mu_1, \sigma_0^2, \sigma_1^2) = \sum_{k=0}^{1} \Bigg(\alpha \sum_{(x,y)\in D_{t,k}} \left[-\log \sigma_k - \frac{(x-\mu_k)^2}{2\sigma_k^2} \right] + (1-\alpha) \sum_{(x,y)\in D_{o,k}} \left[-\log \sigma_k - \frac{(x-\mu_k)^2}{2\sigma_k^2} \right] \Bigg) + \text{const.}\,.$$

α is a weight that controls the contribution of the OOD samples in the log-likelihood function. Under the above log-likelihood, the maximum likelihood estimate for μ_k is

$$\hat{\mu}_k = \frac{\alpha \sum_{(x,y)\in D_{t,k}} x + (1-\alpha) \sum_{(x,y)\in D_{o,k}} x}{\alpha |D_{t,k}| + (1-\alpha)|D_{o,k}|}. \tag{B.8}$$

We can make use of the above $\hat{\mu}_k$ to get a weighted FLD decision rule using eq. (B.4).

B.1.4 Deriving the Generalization Error of the Target Task for Synthetic Tasks with Weighted FLD

We consider the synthetic mixture-of-gaussians task with $\sigma^2 = 1$. We re-write $\hat{\mu}_k$ from eq. (B.8) using notation from section B.1.2:

$$\hat{\mu}_k = \frac{n\alpha \bar{x}_{t,k} + m(1-\alpha)\bar{x}_{o,k}}{n\alpha + m(1-\alpha)}.$$

We can explicitly compute $\bar{x}_{t,k}$ and $\bar{x}_{o,k}$ in the OOD-aware setting since we can separate target samples from OOD samples. For the synthetic dataset, the threshold $\hat{c}_\alpha = \frac{\hat{\mu}_0 + \hat{\mu}_1}{2}$ of the hypothesis $\hat{h}_\alpha$ follows a normal distribution $\mathcal{N}(\mu_{h\alpha}, \sigma_{h\alpha}^2)$ where

$$\mu_{h\alpha} = \mathbb{E}[\hat{c}_\alpha] = \frac{m(1-\alpha)\Delta}{n\alpha + m(1-\alpha)}$$
$$\sigma_{h\alpha}^2 = \text{Var}[\hat{c}_\alpha] = \frac{\alpha^2 n + (1-\alpha)^2 m}{(\alpha n + (1-\alpha)m)^2}$$

Similar to the section B.1.2, we derive an analytical expression for the expected target risk of the weighted FLD, which is

$$e_t(\hat{h}_\alpha) = \frac{1}{2}\left[\Phi\left(\frac{\mu_{h\alpha} - \mu}{\sqrt{1+\sigma_{h\alpha}^2}} \right) + \Phi\left(\frac{-\mu_{h\alpha} - \mu}{\sqrt{1+\sigma_{h\alpha}^2}} \right) \right] \tag{B.9}$$

B.2 Tools for visualizing trajectories of representations

B.2.1 Bhattacharyya Distance

We provide additional details regarding eq. (4.2). Let $\bar{y} = (y_1, \cdots y_N)$, denote the labels assigned to each of the N samples. Since there are C classes in total, $\vec{y}$ can take a total of C^N different values denoted by the set Y^N. Given, two models P_u and P_v, the Bhattacharyya distance averaged over the samples is

$$
\begin{aligned}
\mathrm{d_B}(P_u, P_v) &:= -N^{-1} \log\left(\sum_{\vec{y} \in Y^N} \sqrt{P_u(\bar{y}) P_v(\bar{y})} \right) \\
&= -N^{-1} \log\left(\sum_{\vec{y} \in Y^N} \prod_{n=1}^{N} \sqrt{p_u(y_n)\, p_v(y_n)} \right) \\
&= -N^{-1} \log\left(\sum_{y_1=1}^{C} \sum_{y_2=1}^{C} \cdots \sum_{y_N=1}^{C} \left(\prod_{n=1}^{N} \sqrt{p_u(y_n)\, p_v(y_n)} \right) \right) \\
&= -N^{-1} \log\left(\prod_{i=1}^{N} \left(\sum_{y_i=1}^{C} \sqrt{p_u(y_i)\, p_v(y_i)} \right) \right) \\
&= -N^{-1} \sum_{i=1}^{N} \log\left(\sum_{y_i=1}^{C} \sqrt{p_u(y_i)\, p_v(y_i)} \right).
\end{aligned}
$$

Uncovering the structure of high-dimensional probabilistic models is difficult because most distances between probability distributions saturate with the dimensionality, e.g., the Hellinger distance which is a metric, is essentially equal to 2 in high-dimensions. Quinn et al. (2019b, Figure 1) illustrates how a high-dimensional model benefits from using the Bhattacharyya distance compared to using the Hellinger distance in uncovering the intrinsic structure of the manifold. We believe that the logarithm in the Bhattacharyya distance keeps it well-behaved. We actually know of one other distance that gives meaningful results and that is the symmetric KL-divergence (Teoh et al., 2020), for the same reason: due to the logarithm. All analysis in our paper can therefore be done with the symmetric-KL divergence (which is also not a metric) and the results do look similar.

There are a couple more reasons that motivated us to use the Bhattacharyya distance. First, the Bhattacharyya

distance to the truth P_* is equal to one half of the cross-entropy loss. Second, it is reassuring that both the Bhattacharyya distance locally gives the Fisher Information Matrix, which is positive semi-definite and therefore induces a local metric.

Bhattacharyya distance violates the triangle inequality and we speculate that this is necessary in order to uncover the low-dimensional structure in high-dimensional data. Understanding why it is important to violate the triangle inequality is a deep question and we do not know how to answer it yet. We do not use the Bhattacharyya distance itself to say things like "task A is close to task B" and as a result, the conclusions do not suffer from the violation of the triangle inequality. We only say things like "training on task A is equivalent to training for 80% progress on task B, or "training using contrastive learning is equivalent to training using supervised learning for 25% of the progress".

B.2.2 Imprinting as an alternative to training the final layer

Consider a total of C classes. We would like to find weights $\{w_c\}_{c=1}^{C}$ that maximize the log-probability of the samples, under the constraint that for all $c \in C$, the norm of the weights $\|w_c\|$ is 1. Let $\varphi(x)$ denote a internal representation of sample x. The log-probability

$$\sum_{x:y_x=c} \log p(y = c \mid x) = \sum_{x:y_x=c} w_c \cdot \varphi(x) - \sum_{x:y_x=c} \log\left(\sum_{j=c}^{C} \exp\left(w_c \cdot \varphi(x)\right)\right), \tag{B.10}$$

is proportional to the inner-product $w_c \cdot \sum_{x:y_x=c} \varphi(x)$. Maximizing just this term under the norm constraint, we get the imprinted weights $\sum_i \phi(x_i^c)/\|\sum_i \phi(x_i^c)\|$ as the solution. Deriving an analytical expression for the optimal value of $\{w_c\}_{i=1}^{n_c}$ is difficult and hence we use the imprinted weights as an approximate solution. In our experiments, we found that the imprinted weights achieve an accuracy close to the optimal weights while being significantly easier to compute.

B.2.3 Invariant transformations of the internal representation

The internal representations are invariant to orthogonal transformations provided that we use imprinting to define a probabilistic model. This is because the internal representations define the same probabilistic model ever after an orthogonal transformation. Consider two internal representation ϕ and $U \cdot \phi$ where U is an

orthogonal matrix. We note that the probabilistic model for $U \cdot \phi$ after imprinting is

$$\begin{aligned}\log p_2(y = c \mid x_i) &= \frac{U \cdot \sum_{y_x=c} \phi(x)}{\| U \cdot \sum_{y_x=c} \phi(x)\|} \cdot (U \cdot \phi(x_i)) - \log\left(\sum_{c=1}^{C} \exp\left(\frac{U \cdot \sum_{y_x=c} \phi(x)}{\| U \cdot \sum_{y_x=c} \phi(x)\|} \cdot (U \cdot \phi(x_i))\right)\right) \\ &= \frac{\sum_{y_x=c} \phi(x)}{\| \sum_{y_x=c} \phi(x)\|} \cdot \phi(x_i) - \log\left(\sum_{c=1}^{C} \exp\left(\frac{\sum_{y_x=c} \phi(x)}{\| \sum_{y_x=c} \phi(x)\|} \cdot \phi(x_i)\right)\right).\end{aligned}$$

The probabilistic model for the representation $U \cdot \phi$ is identical to the probabilistic model for representation ϕ since norms and angles are preserved under orthogonal transformations. Hence the Bhattacharyya distance between ϕ and $U \cdot \phi$ is zero.

The imprinting procedure can be thought of as removing information from the representation that is not relevant to prediction on a task. While this is true for all datasets in general, there could exist some additional structure in the data that results in more invariances (e.g., more than invariances to orthogonal transformations $O(n)$).

B.2.4 Measuring goodness-of-fit of an InPCA embedding using explained stress

We would like to measure if a k-dimensional sub-space accurately preserves the true distances. For this purpose, we define a quantity called the "explained stress" that estimates the fraction of pairwise distances in the original space that are preserved in the k-dimensional embedding. This is analogous to the explained variance in principal component analysis (PCA); but explained variance is a measure of the how well the original points are preserved in the embedding whereas explained stress approximates how well pairwise Bhattacharyya distances are preserved. If we consider the embedding to be given by first k eigen-vectors, then the explained stress (χ_k) is

$$\chi_k = 1 - \frac{\left\| W - \sum_{i=1}^{k} \Sigma_{ii}\, U_i U_i^\top \right\|_{\mathrm{F}}}{\|W\|_{\mathrm{F}}} = 1 - \sqrt{\frac{\sum_{i=k+1}^{m} \Sigma_{ii}^2}{\sum_{i=1}^{m} \Sigma_{ii}^2}}. \tag{B.11}$$

Note that InPCA finds an embedding that *exactly* maximizes χ_k.

B.2.5 Calculating mean trajectories

We defined the distance between two trajectories to be $\mathrm{d}_{\mathrm{traj}}(\tau_u^{1\to \mathrm{U}}, \tau_v^{2\to \mathrm{U}})$, i.e., the integral of the Bhattacharyya distance between the trajectories after mapping them to the same task and re-indexing them using the geodesic. Say we wish to compare a model trained on two tasks from CIFAR-10: Cats vs. Dogs and Airplane vs. Truck. We initialize multiple models for each of these two supervised learning problems (and we do so for every experiment in this paper) and train these 10 models. We can now calculate the mean trajectory of models on a task

$$\operatorname*{argmin}_{\tau_\mu^1} \frac{1}{K} \sum_{k=1}^{K} \mathrm{d}_{\mathrm{traj}}(\tau_{u_k}^1, \tau_\mu^1).$$

This optimization problem is very challenging because the variable is a trajectory of probabilistic models in a high-dimensional space. Even if we were to split this minimization and do it independently across time, this is still difficult because the solution is the so-called Bhattacharyya centroid on the product manifold defined in eq. (4.1) and cannot be computed in closed form. See (Nielsen and Boltz, 2011) for an iterative formula. We therefore simply take the arithmetic mean of the probability distributions, i.e., $P_{\mu(t)} = \frac{1}{K} \sum_{k=1}^{K} P_{w_i(t)}$. This is similar to ensembling. We use the radius of the tube around the mean trajectory, i.e.,

$$r_u = \max_k \mathrm{d}_{\mathrm{traj}}(\tau_{u_k}^1, \tau_\mu^1)$$

to normalize distances (more precisely, we normalize using the average of the radii of the two trajectories being compared). Note that this radius depends upon time (and is computed after mapping and reindexing the trajectories). If the distance between the means of two sets of trajectories is smaller than their individual average radii, then the tubes around the means intersect each other. In such cases, one can say that the representations learned (at that time point) are not distinguishable. We next show all distances between reindexed points along the trajectories discussed in figs. 4.1, 4.7 and 4.8. Note that each curve gives the integrands in eq. (4.5), not the integral.

BIBLIOGRAPHY

Michael C. Abbott and Benjamin B. Machta. A Scaling Law From Discrete to Continuous Solutions of Channel Capacity Problems in the Low-Noise Limit. *Journal of Statistical Physics*, 176(1):214–227, July 2019. ISSN 1572-9613. doi: 10.1007/s10955-019-02296-2.

Alessandro Achille and Stefano Soatto. Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research*, 19(1):1947–1980, 2018.

Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6430–6439, 2019a.

Alessandro Achille, Glen Mbeng, and Stefano Soatto. Dynamics and Reachability of Learning Tasks. *arXiv:1810.02440 [cs, stat]*, 2019b.

Deepak Agarwal, Lihong Li, and Alexander Smola. Linear-time estimators for propensity scores. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 93–100, 2011.

Alexander A Alemi. Variational predictive information bottleneck. In *Symposium on Advances in Approximate Bayesian Inference*, pages 1–6. PMLR, 2020.

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3366–3375, 2017.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154, 2018.

Noga Alon, Shai Ben-David, Nicolò Cesa-Bianchi, and David Haussler. Scale-sensitive dimensions, uniform convergence, and learnability. *Journal of the ACM*, 44(4):615–631, July 1997.

Shun-ichi Amari. *Information Geometry and Its Applications*, volume 194 of *Applied Mathematical Sciences*. Springer, Tokyo, 2016.

Arash Amini, Richard Baumgartner, and Dai Feng. Target alignment in truncated kernel ridge regression. *Advances in Neural Information Processing Systems*, 35:21948–21960, 2022.

Suguru Arimoto. An algorithm for computing the capacity of arbitrary discrete memoryless channels. *IEEE Transactions on Information Theory*, 18(1):14–20, 1972.

Martin Arjovsky. *Out of Distribution Generalization in Machine Learning*. PhD thesis, New York University,

2020.

Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332. PMLR, 2019.

Joseph J Atick and A Norman Redlich. What does the retina know about natural scenes? *Neural computation*, 4(2):196–210, 1992.

Fred Attneave. Some informational aspects of visual perception. *Psychological review*, 61(3):183, 1954.

Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *Advances in neural information processing systems*, 32, 2019.

David Harold Bailey. Sequential schemes for classifying and predicting ergodic processes. *Stanford University ProQuest Dissertations Publishing*, 1976.

Vijay Balasubramanian. Heterogeneity and Efficiency in the Brain. *Proceedings of the IEEE*, 103(8):1346–1358, 2015.

Vijay Balasubramanian and Peter Sterling. Receptive fields and functional architecture in the retina. *The Journal of physiology*, 587(12):2753–2767, 2009.

Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.

Randall Balestriero and Yann LeCun. How learning by reconstruction produces uninformative features for perception. In *International Conference on Machine Learning*, 2024.

Horace Barlow. Redundancy reduction revisited. *Network: computation in neural systems*, 12(3):241, 2001.

Horace B Barlow. Single units and sensation: A neuron doctrine for perceptual psychology? *Perception*, 1 (4):371–394, 1972.

Horace B Barlow. Unsupervised learning. *Neural computation*, 1(3):295–311, 1989.

Horace B Barlow et al. Possible principles underlying the transformation of sensory messages. *Sensory communication*, 1(01):217–233, 1961.

Adam B Barrett. Exploration of synergistic and redundant information sharing in static and dynamical Gaussian systems. *Physical Review E*, 91(5):052802, 2015.

Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070, 2020.

Jonathan Baxter. Learning internal representations. In *Proceedings of the eighth annual conference on Computational learning theory*, pages 311–320, 1995.

Jonathan Baxter. Theoretical models of learning to learn. In *Learning to learn*, pages 71–94. Springer, 1998.

Jonathan Baxter. A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198, 2000.

Sören Becker, Johanna Vielhaben, Marcel Ackermann, Klaus-Robert Müller, Sebastian Lapuschkin, and Wojciech Samek. Audiomnist: Exploring explainable artificial intelligence for audio analysis on a simple benchmark. *Journal of the Franklin Institute*, 361(1):418–428, 2024.

Suzanna Becker and Geoffrey E Hinton. Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355(6356):161–163, 1992.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R Devon Hjelm. Mine: Mutual information neural estimation. *arXiv preprint arXiv:1801.04062*, 2018.

Shai Ben-David and Reba Schuller Borbely. A notion of task relatedness yielding provable multiple-task learning guarantees. *Machine learning*, 73(3):273–287, 2008.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010.

Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. On the optimization of a synaptic learning rule. In *Preprints Conf. Optimality in Artificial and Biological Neural Networks*, pages 6–8, 1992.

James O Berger, Jos M Bernardo, and Manuel Mendoza. *On Priors That Maximize Expected Information*. Purdue University. Department of Statistics, 1988.

James O Berger, José M Bernardo, and Dongchu Sun. The formal definition of reference priors. *The Annals of Statistics*, 37(2):905–938, 2009.

Pietro Berkes and Laurenz Wiskott. Slow feature analysis yields a rich repertoire of complex cell properties. *Journal of vision*, 5(6):9–9, 2005.

Jose M Bernardo. Reference posterior distributions for Bayesian inference. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(2):113–128, 1979.

David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *arXiv preprint arXiv:1911.09785*, 2019a.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel.

MixMatch: A holistic approach to semi-supervised learning. *Advances in Neural Information Processing Systems*, 32, 2019b.

Anil Bhattacharyya. On a measure of divergence between two multinomial populations. *Sankhyā: the indian journal of statistics*, pages 401–406, 1946.

William Bialek, Ilya Nemenman, and Naftali Tishby. Predictability, complexity, and learning. *Neural computation*, 13(11):2409–2463, 2001.

José M. Bioucas-Dias and Mário A. T. Figueiredo. Multiplicative noise removal using variable splitting and constrained optimization. *IEEE Transactions on Image Processing*, 19:1720–1730, 2009.

Richard Blahut. Computation of channel capacity and rate-distortion functions. *IEEE transactions on Information Theory*, 18(4):460–473, 1972.

Adam Block, Alexander Rakhlin, and Abhishek Shetty. On the performance of empirical risk minimization with smoothed data, 2024.

Ben Blum-Smith and Soledad Villar. Machine learning and invariant theory. *arXiv preprint arXiv:2209.14991*, 2023.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965, 1989.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *arXiv:1606.04838*, 2016.

Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. Introduction to statistical learning theory. In *Summer School on Machine Learning*, pages 169–207. Springer, 2003.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Albert S. Bregman. *Auditory Scene Analysis: The Perceptual Organization of Sound*. The MIT Press, 05 1990. ISBN 9780262269209. doi: 10.7551/mitpress/1486.001.0001.

Richard P. Brent. An algorithm with guaranteed convergence for finding a zero of a function. *The computer journal*, 14(4):422–425, 1971.

William L Briggs et al. *The DFT: An Owners' Manual for the Discrete Fourier Transform*, volume 45. SIAM, 1995.

Geoffrey J Burton and Ian R Moorhead. Color and spatial structure in natural scenes. *Applied optics*, 26(1): 157–170, 1987.

S. Butterworth. On the theory of filter amplifiers. *Experimental Wireless and the Wireless Engineer*, 7:

536–541, 1930.

Jonathon Byrd and Zachary Lipton. What is the effect of importance weighting in deep learning? In *International Conference on Machine Learning*, pages 872–881, 2019.

Shuhao Cao, Peng Xu, and David A. Clifton. How to understand masked autoencoders. *ArXiv*, abs/2202.03670, 2022.

Yuan Cao, Zhiying Fang, Yue Wu, Ding-Xuan Zhou, and Quanquan Gu. Towards understanding the spectral bias of deep learning. *JCAI-21*, 2021.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Giovanni Cavallanti, Nicolo Cesa-Bianchi, and Claudio Gentile. Linear algorithms for online multitask classification. *The Journal of Machine Learning Research*, 11:2901–2934, 2010.

Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

Kenneth Chaney, Fernando Cladera, Ziyun Wang, Anthony Bisulco, M. Ani Hsieh, Christopher Korpela, Vijay Kumar, Camillo J. Taylor, and Kostas Daniilidis. M3ed: Multi-robot, multi-sensor, multi-environment event dataset. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4016–4023, 2023. doi: 10.1109/CVPRW59228.2023.00419.

Pierre Charbonnier, Laure Blanc-Feraud, Gilles Aubert, and Michel Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. In *Proceedings of 1st international conference on image processing*, volume 2, pages 168–172. IEEE, 1994.

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–547, 2018.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *ICLR*, 2019a.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019b.

Gal Chechik, Amir Globerson, M Anderson, E Young, Israel Nelken, and Naftali Tishby. Group redundancy measures reveal redundancy reduction in the auditory pathway. *Advances in neural information processing systems*, 14, 2001.

Guangyong Chen, Fengyuan Zhu, and Pheng-Ann Heng. An efficient statistical method for image noise level estimation. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 477–485, 2015.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, pages 1597–1607, 2020a.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in Neural Information Processing Systems*, 33:22243–22255, 2020b.

Bertrand S Clarke and Andrew R Barron. Jeffreys' prior is asymptotically least favorable under entropy risk. *Journal of Statistical planning and Inference*, 41(1):37–60, 1994.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223, 2016.

Thomas M. Cover. Open problems in information theory. *IEEE USSR Joint Workshop on Information Theory*, 1975.

Ian Covert and Su-In Lee. Improving kernelshap: Practical shapley value estimation via linear regression. *arXiv preprint arXiv:2012.01536*, 2020.

Trevor F Cox and Michael AA Cox. *Multidimensional scaling*. CRC press, 2000.

Koby Crammer, Michael Kearns, and Jennifer Wortman. Learning from multiple sources. *Journal of Machine Learning Research*, 9(8), 2008.

Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. Large scale fine-grained categorization and domain-specific transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4109–4118, 2018.

Pawel Czyz, Frederic Grabowski, Julia Vogt, Niko Beerenwinkel, and Alexander Marx. Beyond normal: On the evaluation of mutual information estimators. *Advances in Neural Information Processing Systems*, 36, 2024.

Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

H Daume, III. Frustratingly Easy Domain Adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 2007.

Ashwin De Silva, Rahul Ramesh, Pratik Chaudhari, and Joshua T Vogelstein. Prospective Learning: Principled Extrapolation to the Future. In *Proc. of Conference on Lifelong Learning Agents (CoLLAs)*, 2023.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

Yue Deng, Wenxuan Zhang, Sinno Jialin Pan, and Lidong Bing. Multilingual jailbreak challenges in large language models. *arXiv preprint arXiv:2310.06474*, 2023.

Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2169–2176, 2017.

Guneet S Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. In *Proc. of International Conference of Learning and Representations (ICLR)*, 2020.

Natalia Díaz-Rodríguez, Vincenzo Lomonaco, David Filliat, and Davide Maltoni. Don't forget, there is more than forgetting: new metrics for continual learning. *arXiv preprint arXiv:1810.13166*, 2018.

Ronald W DiTullio, Eugenio Piasini, Pratik Chaudhari, Vijay Balasubramanian, and Yale E Cohen. Time as a supervisor: Temporal regularity and auditory object learning. *Frontiers In Computational Neuroscience*, 2023.

Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2051–2060, 2017.

Dawei W Dong and Joseph J Atick. Statistics of natural time-varying images. *Network: computation in neural systems*, 6(3):345, 1995.

Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. *Advances in neural information processing systems*, 27, 2014.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Simon S. Du, Wei Hu, Sham M. Kakade, Jason D. Lee, and Qi Lei. Few-Shot Learning via Learning the Representation, Provably. *arXiv:2002.09434 [cs, math, stat]*, February 2020.

Theodoros Evgeniou, Charles A Micchelli, Massimiliano Pontil, and John Shawe-Taylor. Learning multiple tasks with kernel methods. *Journal of machine learning research*, 6(4), 2005.

Rasool Fakoor, Pratik Chaudhari, Stefano Soatto, and Alexander J Smola. Meta-Q-Learning. In *Proc. of*

International Conference of Learning and Representations (ICLR), 2020.

Rasool Fakoor, Jonas Mueller, Zachary C. Lipton, Pratik Chaudhari, and Alexander J. Smola. Time-Varying Propensity Score to Bridge the Gap between the Past and Present. In *ICLR*, 2024.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3762–3773. PMLR, 2020.

Sebastian Farquhar and Yarin Gal. A unifying bayesian view of continual learning. *arXiv preprint arXiv:1902.06494*, 2019a.

Sebastian Farquhar and Yarin Gal. Towards Robust Evaluations of Continual Learning. *arXiv:1805.09733 [cs, stat]*, June 2019b.

Christoph Feichtenhofer, Yanghao Li, Kaiming He, et al. Masked autoencoders as spatiotemporal learners. *Advances in neural information processing systems*, 35:35946–35958, 2022.

Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, and Daan Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. *arXiv:1701.08734 [cs]*, January 2017.

David J Field. Relations between the statistics of natural images and the response properties of cortical cells. *Josa a*, 4(12):2379–2394, 1987.

David J Field. What is the goal of sensory coding? *Neural computation*, 6(4):559–601, 1994.

DJ Field. Scale-invariance and self-similar 'wavelet'transforms: An analysis of natural scenes and mammalian visual systems. *Wavelets, fractals, and Fourier transforms*, pages 151–193, 1993.

Chris Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. Efficiently identifying task groupings for multi-task learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135, 2017.

Makoto Fukushima, Alex M Doyle, Matthew P Mullarkey, Mortimer Mishkin, and Bruno B Averbeck. Distributed acoustic cues for caller identity in macaque vocalization. *Royal Society open science*, 2(12): 150432, 2015.

Joao Gama, Raquel Sebastiao, and Pedro Pereira Rodrigues. On evaluating stream learning algorithms. *Machine learning*, 90:317–346, 2013.

Shuyang Gao, Greg Ver Steeg, and Aram Galstyan. Efficient estimation of mutual information for strongly dependent variables. In *Artificial Intelligence and Statistics*, pages 277–286, 2015.

Yansong Gao and Pratik Chaudhari. A free-energy principle for representation learning. In *Proc. of International Conference of Machine Learning (ICML)*, 2020.

Yansong Gao and Pratik Chaudhari. An information-geometric distance on the space of tasks. In *Proc. of International Conference of Machine Learning (ICML)*, 2021.

Matan Gavish and David L. Donoho. The optimal hard threshold for singular values is 4/sqrt(3). *IEEE Transactions on Information Theory*, 60:5040–5053, 2014.

Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4367–4375, 2018.

Yves Grandvalet, Yoshua Bengio, et al. Semi-supervised learning by entropy minimization. *CAP*, 367: 281–296, 2005.

Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *International Conference on Learning Representations*, 2020.

Yunhui Guo, Mingrui Liu, Tianbao Yang, and Tajana Rosing. Improved schemes for episodic memory-based lifelong learning. In *Advances in Neural Information Processing Systems*, 2020.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

Guy Hacohen and Daphna Weinshall. Principal components bias in over-parameterized linear models, and its manifestation in deep neural networks. *Journal of Machine Learning Research*, 23(155):1–46, 2022.

Nika Haghtalab, Tim Roughgarden, and Abhishek Shetty. Smoothed analysis with adaptive adversaries. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 942–953, 2022. doi: 10.1109/FOCS52979.2021.00095.

Peter J. B. Hancock, Roland J. Baddeley, and Leslie S. Smith. The principal components of natural images. *Network: Computation In Neural Systems*, 3:61–70, 1992.

Steve Hanneke. Learning whenever learning is possible: Universal learning under general stochastic processes. *J. Mach. Learn. Res.*, 22:130:1–130:116, 2021.

Steve Hanneke and Samory Kpotufe. On the value of target data in transfer learning. In *NeurIPS*, 2019.

Steve Hanneke and Samory Kpotufe. A no-free-lunch theorem for multitask learning. *arXiv preprint arXiv:2006.15785*, 2020.

David Haussler. Decision theoretic generalizations of the pac model for neural net and other learning applications. *Information and computation*, 100(1):78–150, 1992.

Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16*, pages 466–483. Springer, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Doll'ar, and Ross B. Girshick. Masked autoencoders are scalable vision learners. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15979–15988, 2021.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

Junlin Hu, Jiwen Lu, and Yap-Peng Tan. Deep transfer metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 325–333, 2015.

Yu Huang, Zixin Wen, Yuejie Chi, and Yingbin Liang. How Transformers Learn Diverse Attention Correlations in Masked Vision Pretraining, June 2024. arXiv:2403.02233.

Aapo Hyvärinen, Jarmo Hurri, and Patrik O. Hoyer. Natural image statistics - a probabilistic approach to early computational vision. In *Computational Imaging and Vision*, 2009.

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data. *arXiv preprint arXiv:2202.00622*, 2022.

Sosuke Ito and Andreas Dechant. Stochastic time evolution, information geometry, and the Cramér-Rao bound. *Physical Review X*, 10(2):021056, 2020.

Tommi Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems*, pages 487–493, 1999.

Saachi Jain, Hadi Salman, Alaa Khaddaj, Eric Wong, Sung Min Park, and Aleksander Madry. A data-based perspective on transfer learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3613–3622, 2023.

Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, et al. *An introduction to statistical learning*, volume 112. Springer, 2013.

Zahra Kadkhodaie and Eero P. Simoncelli. Stochastic solutions for linear inverse problems using the prior implicit in a denoiser. In *Neural Information Processing Systems*, 2021.

David Kane, Phillip Guan, and Martin S Banks. The limits of human stereopsis in space and time. *Journal of Neuroscience*, 34(4):1397–1408, 2014.

Gal Kaplun, Nikhil Ghosh, Saurabh Garg, Boaz Barak, and Preetum Nakkiran. Deconstructing distributions: A pointwise framework of learning. *arXiv preprint arXiv:2202.09931*, 2022.

Prakhar Kaushik, Alex Gain, Adam Kortylewski, and Alan Yuille. Understanding catastrophic forgetting and remembering in continual learning with optimal relevance mapping. *arXiv preprint arXiv:2102.11343*, 2021.

Michael Kearns. Thoughts on hypothesis boosting. *Unpublished manuscript*, 45:105, 1988.

Byoungjip Kim, Jinho Choo, Yeong-Dae Kwon, Seongho Joe, Seungjai Min, and Youngjune Gwon. Self-match: Combining contrastive self-supervision and consistency for semi-supervised learning. *arXiv preprint arXiv:2101.06480*, 2021.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1920–1929, 2019.

Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big Transfer (BiT): General Visual Representation Learning. *arXiv:1912.11370 [cs]*, 2020.

Lingjing Kong, Martin Q. Ma, Guan-Hong Chen, Eric P. Xing, Yuejie Chi, Louis-Philippe Morency, and Kun Zhang. Understanding masked autoencoders via hierarchical latent variable models. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7918–7928, 2023.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMLR, 2019.

Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical review E*, 69(6):066138, 2004.

A. Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. PhD thesis, Computer Science, University of Toronto, 2009.

Abhishek Kumar and Hal Daume III. Learning task grouping and overlap in multi-task learning. *arXiv preprint arXiv:1206.6417*, 2012.

Vitaly Kurin, Alessandro De Palma, Ilya Kostrikov, Shimon Whiteson, and Pawan K Mudigonda. In defense

of the unitary scalarization for deep multi-task learning. *Advances in Neural Information Processing Systems*, 35:12169–12183, 2022.

Peter E Latham and Sheila Nirenberg. Synergy, redundancy, and independence in population codes, revisited. *Journal of Neuroscience*, 25(21):5195–5206, 2005.

Simon Laughlin. A simple coding procedure enhances a neuron's information capacity. *Zeitschrift für Naturforschung c*, 36(9-10):910–912, 1981.

Guillaume Leclerc, Andrew Ilyas, Logan Engstrom, Sung Min Park, Hadi Salman, and Aleksander Madry. Ffcv: Accelerating training by removing data bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12011–12020, June 2023.

Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896, 2013.

Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. *Advances in Neural Information Processing Systems*, 30: 4652–4662, 2017.

Michael S Lewicki. Efficient coding of natural sounds. *Nature neuroscience*, 5(4):356–363, 2002.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5542–5550, 2017.

Hao Li, Pratik Chaudhari, Hao Yang, Michael Lam, Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Rethinking the hyper-parameters for fine-tuning. In *Proc. of International Conference of Learning and Representations (ICLR)*, 2020a.

Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-Tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. A system for massively parallel hyperparameter tuning. *Proceedings of machine learning and systems*, 2:230–246, 2020b.

Yanyu Li, Ju Hu, Yang Wen, Georgios Evangelidis, Kamyar Salahi, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. Rethinking vision transformers for mobilenet size and speed. In *Proceedings of the IEEE international conference on computer vision*, 2023.

Yingzhen Li and Richard E. Turner. R\'enyi Divergence Variational Inference. *arXiv:1602.02311 [cs, stat]*, October 2016.

Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. Temporal Fusion Transformers for interpretable

multi-horizon time series forecasting. *International journal of forecasting*, 37(4):1748–1764, October 2021.

Etai Littwin, Omid Saremi, Madhu Advani, Vimal Thilak, Preetum Nakkiran, Chen Huang, and Joshua Susskind. How jepa avoids noisy features: The implicit bias of deep linear self distillation networks. *arXiv preprint arXiv:2407.03475*, 2024.

Hao Liu and Huaping Liu. Continual learning with recursive gradient optimization. *arXiv preprint arXiv:2201.12522*, 2022.

Xialei Liu, Chenshen Wu, Mikel Menta, Luis Herranz, Bogdan Raducanu, Andrew D Bagdanov, Shangling Jui, and Joost van de Weijer. Generative feature replay for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 226–227, 2020.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6470–6479, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

Richard F Lyon, Andreas G Katsiamis, and Emmanuel M Drakakis. History and future of auditory filter models. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 3809–3812. IEEE, 2010.

Siyuan Ma and Mikhail Belkin. Diving into the shallows: A computational perspective on large-scale shallow learning. *Advances in neural information processing systems*, 30, 2017.

Sadhika Malladi, Alexander Wettig, Dingli Yu, Danqi Chen, and Sanjeev Arora. A kernel-based view of language model fine-tuning. *arXiv preprint arXiv:2210.05643*, 2022.

Stephane Mallat. *A Wavelet Tour of Signal Processing: The Sparse Way*. Elsevier, 2008.

Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82, 2018.

Pranshu Malviya, Balaraman Ravindran, and Sarath Chandar. Tag: Task-based accumulated gradients for lifelong learning. *arXiv preprint arXiv:2105.05155*, 2021.

Jialin Mao, Itay Griniasty, Han Kheng Teoh, Rahul Ramesh, Rubing Yang, Mark Transtrum, James P. Sethna, and Pratik Chaudhari. The Training Process of Many Deep Networks Explores the Same Low-Dimensional Manifold. *Proceedings of the National Academy of Sciences (PNAS)*, 2024.

David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. MIT press, 2010.

Charles H Martin and Michael W Mahoney. Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning. *Journal of Machine Learning Research*, 22 (165):1–73, 2021.

Henry H Mattingly, Mark K Transtrum, Michael C Abbott, and Benjamin B Machta. Maximizing the information learned from finite data selects a simple model. *Proceedings of the National Academy of Sciences*, 115(8):1760–1765, 2018.

Andreas Maurer. Bounds for linear multi-task learning. *The Journal of Machine Learning Research*, 7: 117–139, 2006.

Peyman Milanfar and Mauricio Delbracio. Denoising: A powerful building-block for imaging, inverse problems, and machine learning. *ArXiv*, abs/2409.06219, 2024.

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Dilan Gorur, Razvan Pascanu, and Hassan Ghasemzadeh. Linear mode connectivity in multitask and continual learning. *arXiv preprint arXiv:2010.04495*, 2020a.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Razvan Pascanu, and Hassan Ghasemzadeh. Understanding the role of training regimes in continual learning. *arXiv preprint arXiv:2006.06958*, 2020b.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4003, 2016.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

Decebal Constantin Mocanu, Maria Torres Vega, Eric Eaton, Peter Stone, and Antonio Liotta. Online contrastive divergence with generative replay: Experience replay without storing data. *arXiv preprint arXiv:1610.05555*, 2016.

Gusztav Morvai, Sidney Yakowitz, and Laszlo Gyorfi. Nonparametric inference for ergodic, stationary time

series. *The Annals of Statistics*, 24(1):370–379, 1996.

Eric Nalisnick and Padhraic Smyth. Variational reference priors. In *Proc. of International Conference of Learning and Representations (ICLR)*, 2017.

Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational continual learning. In *International Conference on Learning Representations*, 2018.

Frank Nielsen and Sylvain Boltz. The burbea-rao and bhattacharyya centroids. *IEEE Transactions on Information Theory*, 57(8):5455–5466, 2011.

A.B. Nobel. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *IEEE Transactions on Information Theory*, 49(1):83–98, 2003.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

Bruno A Olshausen. Sparse coding of time-varying natural images. In *Proc. of the Int. Conf. on Independent Component Analysis and Blind Source Separation*, volume 2, 2000.

Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision research*, 37(23):3311–3325, 1997.

Alan V Oppenheim. *Discrete-time signal processing*. Pearson Education India, 1999.

D. S. Ornstein. Guessing the next output of a stationary process. *Israel Journal of Mathematics*, 30(3): 292—296, 1978.

Pingbo Pan, Siddharth Swaroop, Alexander Immer, Runa Eschenhagen, Richard Turner, and Mohammad Emtiyaz E Khan. Continual deep learning by functional regularisation of memorable past. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 4453–4464. Curran Associates, Inc., 2020.

Namuk Park, Wonjae Kim, Byeongho Heo, Taekyung Kim, and Sangdoo Yun. What Do Self-Supervised Vision Transformers Learn?, May 2023.

Liangzu Peng, Paris Giampouras, and Rene Vidal. The ideal continual learner: An agent that never forgets. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1406–1415, 2019.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word represen-

tation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

A Pentina and C H Lampert. Lifelong learning with non-iid tasks. *Adv. Neural Inf. Process. Syst.*, 2015.

Leon N Piotrowski and Fergus William Campbell. A demonstration of the visual importance and flexibility of spatial-frequency amplitude and phase. *Perception*, 11:337 – 346, 1982.

David Pollard. *Convergence of stochastic processes*. Springer Science & Business Media, 2012.

Phillip Pope, Chen Zhu, Ahmed Abdelkader, Micah Goldblum, and Tom Goldstein. The intrinsic dimension of images and its impact on learning. *arXiv preprint arXiv:2104.08894*, 2021.

Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *European conference on computer vision*, pages 524–540. Springer, 2020.

Stanislav Pyatykh, Jürgen W. Hesser, and Lei Zheng. Image noise level estimation by principal component analysis. *IEEE Transactions on Image Processing*, 22:687–699, 2013.

Hang Qi, Matthew Brown, and David G Lowe. Low-shot learning with imprinted weights. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5822–5830, 2018.

Joaquin Quiñonero-Candela, editor. *Dataset Shift in Machine Learning*. Neural Information Processing Series. Mit Press, Cambridge, Mass, 2009. ISBN 978-0-262-17005-5.

Katherine N. Quinn, Colin B. Clement, Francesco De Bernardis, Michael D. Niemack, and James P. Sethna. Visualizing probabilistic models and data with intensive principal component analysis. *Proceedings of the National Academy of Sciences*, 116(28):13762–13767, 2019a. ISSN 0027-8424. doi: 10.1073/pnas.1817218116.

Katherine N Quinn, Colin B Clement, Francesco De Bernardis, Michael D Niemack, and James P Sethna. Visualizing probabilistic models and data with intensive principal component analysis. *Proceedings of the National Academy of Sciences*, 116(28):13762–13767, 2019b.

Katherine Nicholson Quinn. *Patterns of Structural Hierarchies in Complex Systems*. PhD thesis, Cornell University, 2019.

Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? In *Neural Information Processing Systems*, 2021.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *ICML*, 2019.

Meisam Rakhshanfar and Maria A. Amer. Estimation of gaussian, poissonian–gaussian, and processed visual

noise and its level function. *IEEE Transactions on Image Processing*, 25(9):4172–4185, 2016. doi: 10.1109/TIP.2016.2588320.

Rahul Ramesh and Pratik Chaudhari. Model zoo: A growing brain that learns continually. In *International Conference on Learning Representations*, 2022.

Rahul Ramesh, Anthony Bisulco, Ronald W DiTullio, Linran Wei, Vijay Balasubramanian, Kostas Daniilidis, and Pratik Chaudhari. Many perception tasks are highly redundant functions of their input data. *arXiv preprint arXiv:2407.13841*, 2024.

J. Rapin and O. Teytaud. Nevergrad - A gradient-free optimization platform, 2018.

Charles P Ratliff, Bart G Borghuis, Yen-Hong Kao, Peter Sterling, and Vijay Balasubramanian. Retina is structured to process an excess of darkness in natural scenes. *Proceedings of the National Academy of Sciences*, 107(40):17368–17373, 2010.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 506–516, 2017.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.

Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, 2015.

Clemens Rosenbaum, Tim Klinger, and Matthew Riemer. Routing networks: Adaptive selection of non-linear functions for multi-task learning. *arXiv preprint arXiv:1711.01239*, 2017.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Boris Yakovlevich Ryabko. Prediction of random sequences and universal coding. *Problems of Information Theory*, 24(2):87–96, 1988.

Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Mutual exclusivity loss for semi-supervised deep learning. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 1908–1912. IEEE, 2016.

Robert E Schapire. The strength of weak learnability. *Machine learning*, 5:197–227, 1990.

Robert E Schapire and Yoav Freund. *Boosting: Foundations and Algorithms*. Emerald Group Publishing Limited, 2013.

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4548–4557. PMLR, 2018.

Lloyd S Shapley et al. A value for n-person games. *Contribution to the Theory of Games*, 2, 1953.

Cong Shi, Shrinivas J. Pundlik, and Gang Luo. Without low spatial frequencies, high resolution vision would be detrimental to motion perception. *Journal of Vision*, 20, 2020.

Xing Shi, Inkit Padhi, and Kevin Knight. Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, 2016.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 2994–3003, 2017.

Eero P Simoncelli and Bruno A Olshausen. Natural image statistics and neural representation. *Annual review of neuroscience*, 24(1):1193–1216, 2001.

Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of residual networks using large learning rates. *ArXiv*, abs/1708.07120, 2017.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. FixMatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in Neural Information Processing Systems*, 33, 2020.

Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning*, pages 9120–9132. PMLR, 2020.

Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv preprint arXiv:2106.10270*, 2021.

Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. Adashare: Learning what to share for efficient deep multi-task learning. *arXiv preprint arXiv:1911.12423*, 2019.

Sonia Tabibi, Andrea Kegel, Wai Kong Lai, and Norbert Dillier. Investigating the use of a gammatone filterbank for a cochlear implant coding strategy. *Journal of neuroscience methods*, 277:63–74, 2017.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33:7537–7547, 2020.

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv preprint arXiv:1703.01780*, 2017.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Han Kheng Teoh, Katherine N. Quinn, Jaron Kent-Dobias, Colin B. Clement, Qingyang Xu, and James P. Sethna. Visualizing probabilistic models in Minkowski space with intensive symmetrized Kullback-Leibler embedding. *Physical Review Research*, 2(3):033221, August 2020. ISSN 2643-1564.

Mitchell G. A. Thomson, David H. Foster, and Robert J. Summers. Human sensitivity to phase perturbations in natural images: A statistical framework. *Perception*, 29:1057 – 1069, 2000.

Sebastian Thrun. Lifelong learning algorithms. In *Learning to learn*, pages 181–209. Springer, 1998.

Sebastian Thrun and Tom M Mitchell. Lifelong robot learning. *Robotics and autonomous systems*, 15(1-2): 25–46, 1995.

Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. In *Learning to learn*, pages 3–17. Springer, 1998.

Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. In *Proc. of the 37-Th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377, 1999.

Michalis K. Titsias, Jonathan Schwarz, Alexander G. de G. Matthews, Razvan Pascanu, and Yee Whye Teh. Functional regularisation for continual learning with gaussian processes. In *International Conference on Learning Representations*, 2020.

Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems*, 35: 10078–10093, 2022.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bash-

lykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Nilesh Tripuraneni, Chi Jin, and Michael I Jordan. Provable meta-learning of linear representations. *arXiv preprint arXiv:2002.11684*, 2020a.

Nilesh Tripuraneni, Michael Jordan, and Chi Jin. On the theory of transfer learning: The importance of task diversity. *Advances in Neural Information Processing Systems*, 33, 2020b.

Gido M Van de Ven, Tinne Tuytelaars, and Andreas S Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022.

van A Van der Schaaf and JH van van Hateren. Modelling the power spectra of natural images: Statistics and information. *Vision research*, 36(17):2759–2770, 1996.

Johannes H van Hateren. A theory of maximizing sensory information. *Biological cybernetics*, 68(1):23–29, 1992.

Grant Richard Van Horn. *Towards a Visipedia: Combining Computer Vision and Communities of Experts*. PhD thesis, California Institute of Technology, 2019.

Simon Vandenhende, Stamatios Georgoulis, Bert De Brabandere, and Luc Van Gool. Branched multi-task networks: Deciding what layers to share. *arXiv preprint arXiv:1904.02920*, 2019.

Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1999.

Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity: festschrift for alexey chervonenkis*, pages 11–30. Springer, 2015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Ramakrishna Vedantam, David Lopez-Paz, and David J Schwab. An Empirical Investigation of Domain Generalization with Empirical Risk Minimizers. In *Advances in Neural Information Processing Systems*, volume 34, pages 28131–28143. Curran Associates, Inc., 2021.

Gido M Ven, Hava T Siegelmann, Andreas S Tolias, et al. Brain-inspired replay for continual learning with artificial neural networks. *Nature Communications*, 11(1):1–14, 2020.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in Neural Information Processing Systems*, 29:3630–3638, 2016.

Joshua T Vogelstein, Jayanta Dey, Hayden S Helm, Will LeVine, Ronak D Mehta, Ali Geisa, Gido M van de Ven, Emily Chang, Chenyu Gao, Weiwei Yang, et al. Omnidirectional transfer for quasilinear lifelong learning. *arXiv preprint arXiv:2004.12908*, 2020a.

Joshua T Vogelstein, Jayanta Dey, Hayden S Helm, Will LeVine, Ronak D Mehta, Tyler M Tomita, Haoyin Xu, Ali Geisa, Qingyang Wang, Gido M van de Ven, Chenyu Gao, Weiwei Yang, Bryan Tower, Jonathan Larson, Christopher M White, and Carey E Priebe. A simple lifelong learning approach. *arXiv [cs.AI]*, April 2020b.

Limin Wang, Bingkun Huang, Zhiyu Zhao, Zhan Tong, Yinan He, Yi Wang, Yali Wang, and Yu Qiao. Videomae v2: Scaling video masked autoencoders with dual masking. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14549–14560, 2023.

Alexander Wei, Wei Hu, and Jacob Steinhardt. More than a toy: Random matrix models predict how real-world neural representations generalize. *arXiv preprint arXiv:2203.06176*, 2022a.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022b.

Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. *arXiv preprint arXiv:2002.06715*, 2020.

Felix Wichmann, Doris I. Braun, and Karl R. Gegenfurtner. Phase noise and the classification of natural images. *Vision Research*, 46:1520–1529, 2006.

Paul L Williams and Randall D Beer. Nonnegative decomposition of multivariate information. *arXiv preprint arXiv:1004.2515*, 2010.

Laurenz Wiskott and Terrence J. Sejnowski. Slow Feature Analysis: Unsupervised Learning of Invariances. *Neural Computation*, 14(4):715–770, April 2002. ISSN 0899-7667.

John Wright and Y. Ma. *High-dimensional data analysis with low-dimensional models: Principles, computation, and applications*. Cambridge University Press, 2022.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36:69798–69818, 2023a.

Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: a simple framework for masked image modeling. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9643–9653, 2021.

Zhenda Xie, Zigang Geng, Jingcheng Hu, Zheng Zhang, Han Hu, and Yue Cao. Revealing the Dark Secrets of Masked Image Modeling. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14475–14485, Vancouver, BC, Canada, June 2023b. IEEE. ISBN 9798350301298.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International conference on machine learning*, pages 10524–10533. PMLR, 2020.

Ju Xu and Zhanxing Zhu. Reinforced continual learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 907–916, 2018.

Yi Xu, Lei Shang, Jinxing Ye, Qi Qian, Yu-Feng Li, Baigui Sun, Hao Li, and Rong Jin. Dash: Semi-supervised learning with dynamic thresholding. In *International Conference on Machine Learning*, pages 11525–11536. PMLR, 2021.

Rubing Yang, Jialin Mao, and Pratik Chaudhari. Does the data induce capacity control in deep learning? In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 25166–25197. PMLR, 17–23 Jul 2022.

Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, 2007.

Thomas Yerxa, Yilun Kuang, Eero Simoncelli, and SueYeon Chung. Learning efficient coding of natural images with maximum manifold capacity representations. *Advances in Neural Information Processing Systems*, 36:24103–24128, 2023.

Jaehong Yoon, Saehoon Kim, Eunho Yang, and Sung Ju Hwang. Scalable and order-robust continual learning with additive parameter decomposition. *arXiv preprint arXiv:1902.09432*, 2019.

Jaehong Yoon, Wonyong Jeong, Giwoong Lee, Eunho Yang, and Sung Ju Hwang. Federated continual learning with weighted inter-client transfer. In *International Conference on Machine Learning*, pages 12073–12086. PMLR, 2021.

Yaodong Yu, Kwan Ho Ryan Chan, Chong You, Chaobing Song, and Yi Ma. Learning diverse and discriminative representations via the principle of maximal coding rate reduction. *Advances in Neural Information Processing Systems*, 33:9422–9434, 2020.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3712–3722, 2018.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pages 12310–12320, 2021.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR, 2017.

Chen Zeno, Itay Golan, Elad Hoffer, and Daniel Soudry. Task agnostic continual learning using online variational bayes. *arXiv preprint arXiv:1803.10123*, 2018.

Qi Zhang, Yifei Wang, and Yisen Wang. How mask matters: Towards theoretical understandings of masked autoencoders. *ArXiv*, abs/2210.08344, 2022.

Richard Zhang. Making convolutional networks shift-invariant again. In *International conference on machine learning*, pages 7324–7334. PMLR, 2019.

X Zhang, M G Heinz, I C Bruce, and L H Carney. A phenomenological model for the responses of auditory-nerve fibers: I. nonlinear tuning with compression and suppression. *J. Acoust. Soc. Am.*, 109(2):648–670, 2001.

Zhongxin Zhang. *Discrete noninformative priors*. PhD thesis, Yale University, 1994.

Zhao Zhao, Sai-hua Zhang, Zhi-yong Xu, Kristen Bellisario, Nian-hua Dai, Hichem Omrani, and Bryan C Pijanowski. Automated bird acoustic event detection and robust species classification. *Ecological Informatics*, 39:99–108, 2017.

Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5122–5130, 2017.

Zhi-Hua Zhou and Ming Li. Semi-supervised learning by disagreement. *Knowledge and Information Systems*, 24(3):415–439, 2010.