ID N. AIDR02/18800

UNIVERSITÀ CAMPUS BIO-MEDICO DI ROMA

DEPARTMENT OF ENGINEERING

UNIVERSITÀ DEGLI STUDI DI TORINO

DIPARTIMENTO DI FISICA

Italian National Ph.D. in Artificial Intelligence
Health and Life Sciences
XXXVIII Cycle

How Data Organization Shapes Neural Network Learning: Hierarchical Labels and Spectral Structure

Supervisors

Michele Caselle

Piero Fariselli

Paolo Provero

Candidate

Davide Pirovano

Abstract

Visual classification depends on two fundamental factors: image structure (the relationships encoded in pixel values) and categorical organization (the labels indicating what they depict). This thesis explores how these data properties shape neural network training and responses.

First, I examine hierarchical label structures in visual datasets. While classification tasks are defined at a specific granularity, training on finer-grained labels can boost performance. I show that this benefit depends on dataset size, model capacity, and data geometry, revealing when granularity enhances generalization. Through systematic experiments on synthetic and real datasets, I identify the conditions under which hierarchical training provides advantages over flat classification.

Second, I investigate how the power-law spectra of natural images affect neural network sensitivity, testing both standard CNNs and biologically-inspired retinal models. Using synthetic images with controlled spectral exponents, I find that networks across these architectures show peak sensitivity near the characteristic exponent of natural images. I show that this pattern arises from the convolutional structure rather than learning by comparing random convolutional networks to fully connected ones, and find that it depends on the input dimension. Finally, I show that these spectral preferences influence learning, with classification performance varying according to a dataset's spectral properties.

These studies show how data properties influence training strategies and how architectural constraints modulate network sensitivity to image structure. Together, they provide insights into the interplay between data organization and neural network behavior, with implications for both understanding these systems and designing better training procedures.

Acknowledgements

I would like to thank Matteo Osella, who unfortunately, for bureaucratic reasons, could not be among the official supervisors, but who has been the main supervisor of this project. Thank you for the useful advice, for always creating a stimulating and calm environment to work in. The PhD experience under your supervision has been truly great.

I would like to thank Michele Caselle, who brought me into this project and whose input has always been valuable along the way. Thanks also to Piero Fariselli and Paolo Provero for the helpful exchanges.

Finally, I would like to thank Ulisse Ferrari, who supervised me during my time abroad in Paris. Thank you for the scientific support, always sharp and constructive, and for the human side of it too, which meant a lot during some genuinely difficult moments. Thank you, more broadly, for the time you dedicated to me, treating me in every way as one of your own students, despite me only being a visitor.

Contents

List of Figures

Chapter 1

Introduction

1.1 Purpose of the work

This thesis describes the research work carried out during my PhD program at the University of Turin, with a period spent at the Institut de la Vision under Sorbonne Université in Paris. The research focuses on how data organization affects neural network behavior, examining both training dynamics and network activations.

In recent decades, machine learning has proven highly effective across diverse fields, from computer vision to finance and biology, consistently achieving outstanding performance. The field has experienced exponential growth, particularly following the introduction of deep learning architectures. Among the various applications, visual classification has emerged as a particularly successful domain, with neural networks achieving remarkable accuracy in recognizing and categorizing images.

Classification problems, where models must assign data points to predefined categories, represent a fundamental task in machine learning. In computer vision, these tasks range from medical diagnosis—where distinguishing between benign and malignant lesions can have life-saving implications—to autonomous driving systems that must identify pedestrians, vehicles, and road signs. The success of deep learning in these applications has not only demonstrated the power of these methods but has also raised questions about the factors that determine their effectiveness.

Visual classification tasks are built on two fundamental components at the data level. First, there is the visual input itself: images represented as arrays of pixel values, with complex mathematical relationships between these values that encode edges, textures, and higher-level structures. Second, there is the categorical organization represented by labels, which assign semantic meaning to these visual patterns. An image of a cat, for instance, might be labeled at various levels of granularity—as "animal," "mammal," or more specifically as "domestic cat"—reflecting the hierarchical nature of how we organize visual categories. These two aspects—image structure and categorical

organization—form the foundation of visual classification systems, yet their influence on neural network learning remains incompletely understood. This thesis investigates how these data properties shape neural network behavior through two distinct but related projects.

The first project examines how hierarchical label structures affect generalization, revealing the conditions under which training with finer-grained categorical information enhances or hinders performance. This work addresses a practical question faced by practitioners: given a dataset with hierarchical labels, should we train on the finest available granularity or match the label resolution to our target task?

The second project explores how the spectral properties of images affect the behavior of convolutional neural networks and biologically-inspired retinal models. Natural images exhibit characteristic power spectra that fall approximately as $\frac{1}{f^\alpha}$ with α around 2, reflecting the spatial correlations present in visual scenes. By generating synthetic images with controlled spectral slopes, this work examines how network sensitivity varies with these statistics across different architectures, and whether the observed patterns arise from architectural constraints or emerge through learning.

Together, these projects provide different perspectives on how data properties influence neural networks. The first focuses on categorical structure and its interaction with learning, while the second examines the relationship between image statistics and network architecture. Both address how fundamental characteristics of the data shape the behavior of systems designed to process them, offering insights into when and why certain training strategies or architectural choices prove effective.

1.2 Thesis structure

This thesis is organized into three chapters. Chapter 2 presents the investigation of hierarchical label structures, examining when and why training on fine-grained labels enhances or impedes generalization performance in coarse-grained classification tasks. Chapter 3 describes the study of spectral properties in natural images, revealing how convolutional neural networks exhibit sensitivity to power law statistics and whether this sensitivity emerges from architectural constraints or learning. Chapter 4 synthesizes the findings from both projects, discusses their broader implications for understanding neural network behavior, and outlines directions for future research. Additional technical details, supplementary experimental results, and implementation specifics are provided in the appendixes A and B.

Chapter 2

Should we Always Train Models on Fine-Grained Classes?

2.1 Introduction

The general knowledge that we have across various fields is structured hierarchically [1]. In the natural sciences, living organisms are classified from broad kingdoms down to specific species, with multiple intermediate levels of granularity. In medicine, the International Classification of Diseases (ICD) maintained by the World Health Organization organizes diseases from general categories to specific diagnoses. Similar hierarchical structures appear in gene annotation databases like Gene Ontology [2], which assigns functional labels at various levels of specificity.

Visual recognition datasets often reflect this hierarchical organization. ImageNet [3], for instance, arranges its thousands of object classes according to semantic relationships inherited from WordNet. Other large-scale datasets, such as CIFAR-100 [4], preserve similar nested category structures, where specific labels (e.g., crocodile, lizard, snake, turtle) belong to broader groups (e.g., reptiles).

When faced with a classification task defined at a particular level of granularity, a straightforward choice is to match the training labels to that level. If the goal is to separate animals from vehicles, we can label training images with just these two broad classes. Alternatively, we could train the network on finer classes such as cats, dogs, cars, trucks and then obtain coarse predictions by aggregating the fine-grained outputs that belong to the same superclass. Intuitively, it seems unnecessary to learn distinctions between cats and dogs, or between cars and trucks, if the model will never need to make such discriminations. However, evidence from various applications shows that fine-grained training can enhance coarse-level classification. In dermatology, this approach proved to be remarkably effective. Training a deep network to recognize specific disease subtypes within benign and malignant lesions led to outstanding performances,

matching the accuracy of human dermatologists on the binary classification task [5]. Similar improvements appear across standard image classification experiments, suggesting that networks benefit from training or pre-training on fine-grained labels [6, 7, 8]. However, the generality of this result and its underlying mechanisms remain poorly understood.

One explanation is that fine-grained labels force networks to learn features that capture the subtleties necessary to distinguish between refined categories. This might improve generalization by preventing shortcut learning [9]: the tendency of networks to rely on simple solutions or dataset biases. Deep neural networks achieve strong generalization in part by compressing internal representations to retain task-relevant features while eliminating unnecessary information [10, 11, 12]. This compression manifests in the network's tendency to group subclasses within its learned representations, a phenomenon that occurs even without explicit fine-grained supervision and correlates with generalization across different settings [13]. It is therefore natural to imagine that supervised fine-grained training can further boost this compression process, improving generalization. However, an opposing scenario is also plausible. Some features necessary to discriminate fine classes might be irrelevant for the coarse task, introducing unnecessary complexity that hinders learning. Rather than helping the model focus on what matters, fine-grained supervision could introduce distinctions irrelevant to the coarse objective, compromising optimization and potentially leading to overfitting.

In the context of network distillation, where knowledge is transferred from a large model to a smaller one [14], the transfer often works better when considering a large number of labeled classes [15]. This again points toward a potential advantage of fine-grained class descriptions. The explanation is straightforward: labels describing finer class partitions simply contain more information. However, several questions remain open. Under what conditions is this additional information actually useful? Can it instead become a confounding factor, and if so, when? This chapter investigates the factors that determine when fine-grained training helps. We show that training on fine-grained labels does not always improve performance. Rather, its effectiveness depends on the geometric structure of the dataset and the relationship between dataset size and model capacity. We take an approach inspired by statistical physics [16]: we focus on simple models and look for general patterns, rather than using sophisticated state-of-the-art architectures that might obscure fundamental principles.

2.2 Related Works

Training neural networks on fine-grained labels to improve performance on coarse-grained classification tasks has been studied in various settings, but we still lack a complete understanding of when and why it works. In this section I review the most

relevant works.

2.2.1 Fine-Grained Training for Image Classification

Chen et al. [6] provide the most comprehensive early study of label granularity in CNN-based image classification. They demonstrate across multiple datasets (CIFAR-10, CIFAR-100, and ImageNet subsets) that training with fine-grained labels consistently improves both training accuracy and generalization performance on coarse-grained classification tasks. For example, on CIFAR-10, training on 10 fine-grained classes (individual animal/vehicle types) versus 2 coarse classes (animal vs. vehicle) improved test accuracy from 98.42% to 99.20%. More strikingly, they show that fine-grained training enhances data efficiency: on ImageNet subsets (dog vs. cat classification), models trained on fine-grained labels with only 40% of the available data achieved higher accuracy than models trained on the full dataset with coarse labels. They attribute these improvements to networks learning richer feature representations when forced to discriminate between similar classes.

Chen et al.'s results show that fine-grained training is consistently beneficial approach in a hierarchical classes setting. However, their investigation focuses on a specific regime that leaves open questions about the generality of their conclusions. Their experiments employ large network architectures in settings where the coarse classification task itself is relatively well-solved (e.g., achieving 98-99% accuracy on CIFAR-10 coarse labels). In this regime—where networks have abundant capacity and the base task is nearly mastered—fine-grained training consistently provides benefits or, at minimum, does not harm performance. However, such performances are not typical in more complex real-world tasks, where the interaction between capacity constraints and the difficulty of the task may produce different dynamics. They also explore different geometric settings, varying which fine-grained classes are grouped into coarse categories. However, the overparameterized regime tends to mask the influence of these geometric variations. When a network has abundant capacity, it can accommodate almost any label structure, making it difficult to discern when geometry helps versus when it creates unnecessary complexity. Our work extends this analysis to a broader space of conditions by systematically varying all three factors: relationship between data and labels, network capacity, and dataset size. This extended investigation reveals that fine-grained training is not universally beneficial—its advantages emerge or disappear depending on the relationship between model capacity and data availability, and crucially, on the geometric alignment between fine-grained structure and coarse classification boundaries.

2.2.2 Clinical Applications of Hierarchical Classification

Esteva et al. [5] demonstrate how fine-grained training can be crucial for medical diagnosis. They developed a CNN trained on 129,450 clinical images covering 2,032 different skin diseases organized in a hierarchical taxonomy. Critically, they show that training on fine-grained disease categories substantially improves performance on coarser diagnostic tasks compared to training directly on coarse labels. For instance, on a three-way classification task (benign, malignant, and non-neoplastic lesions), a CNN trained directly on these three coarse classes achieved 69.4% accuracy, while a CNN trained on 757 fine-grained disease classes that map to these three categories achieved 72.1% accuracy.

When evaluated against 21 board-certified dermatologists on biopsy-proven images for critical binary diagnostic tasks, the fine-grained trained network matched expert-level performance. For distinguishing malignant keratinocyte carcinomas from benign seborrheic keratoses using 135 test images, the network achieved an area under the curve (AUC) of 0.96, outperforming most individual dermatologists tested. For melanoma classification, which matters because melanomas account for 75% of skin cancer deaths despite being only 5% of cases, the network achieved 0.94 AUC on 130 test images, again performing on par with the average dermatologist.

The clinical success of their hierarchically-trained model underscores the practical importance of understanding fine-grained training dynamics. This work shows that fine-grained medical training can be crucial for diagnostic applications, motivating our systematic investigation of the conditions under which such approaches succeed or fail.

2.2.3 Information Transfer and Knowledge Distillation

Hinton et al. [14] develop an influential technique called knowledge distillation for transferring knowledge from large, cumbersome models into smaller, more deployable ones. The core idea is elegant: instead of training a small model on hard labels (one-hot vectors), train it to match the soft probability distributions produced by a large teacher model. These soft targets carry more information than hard labels because they reveal the teacher's learned similarities between classes. For example, when shown a BMW, a well-trained model might assign very low probability to both "garbage truck" and "carrot," but the fact that "garbage truck" gets higher probability reveals meaningful structure about what the model has learned.

Their results demonstrate the power of this approach. On MNIST, a small network trained normally achieves 146 test errors, while the same network trained via distillation from a large teacher achieves only 74 errors, compared to 67 for the teacher itself. For speech recognition, they show that a single model distilled from an ensemble of 10 models achieves 60.8% frame accuracy, nearly matching the ensemble's 61.1% while

being much faster to deploy. Perhaps most strikingly, they demonstrate that soft targets act as powerful regularizers. When training on only 3% of their speech data, a model trained with hard labels severely overfits and achieves 44.5% test accuracy, while the same model trained with soft targets from a teacher trained on the full dataset achieves 57.0% accuracy.

In developing this framework, Hinton et al. encounter the fine-grained training phenomenon. They observe that when teacher models are trained on many fine-grained classes, the soft probability distributions they produce carry what they call "dark knowledge," the relative probabilities among incorrect classes that communicate meaningful similarity structures. This observation connects to our work, but from a different angle. While Hinton et al. study how soft targets from fine-grained distributions enable knowledge transfer during distillation, we investigate when training directly on fine-grained hard labels helps versus hurts coarse classification performance. Their information-theoretic perspective provides valuable intuition about why fine-grained information might be useful, the idea that fine distinctions capture rich similarity structures. However, the conditions under which this additional information translates to improved coarse classification, and when it might instead waste capacity or create harmful interference, remain uncharacterized. We build on their intuition by showing that the utility of fine-grained information depends on the geometric alignment between subclass structure and coarse boundaries, the relationship between model capacity and task complexity, and the amount of available training data.

2.2.4 Our Contributions

The existing literature establishes that fine-grained training can improve coarse classification across diverse domains, from vision benchmarks to medical diagnosis. However, a systematic understanding of when and why this occurs is missing. Chen et al. demonstrate consistent benefits but focus on overparameterized regimes where geometric effects are masked. Esteva et al. show clinical impact but don't characterize the underlying conditions for success. Hinton et al. provide an information-theoretic framework but don't connect it to practical training scenarios.

Our work addresses these gaps by systematically varying three factors that prior work has examined in isolation: the geometric relationship between data and labels, network capacity, and dataset size. This reveals that fine-grained training is not universally beneficial. Its advantages emerge or disappear depending on whether the network is overparameterized or capacity-constrained, and crucially, on whether the fine-grained label structure aligns with or conflicts with the coarse classification boundaries. We formalize this geometric relationship through boundary coherence measures, connecting the abstract notion of information in fine-grained labels to concrete prop-

erties of learned decision boundaries. This provides both theoretical insight into when fine-grained training helps and practical guidance for deploying hierarchical training strategies in domains where baseline performance is far from saturated.

2.3 Methods

2.3.1 Datasets

We considered the following widely used datasets for classification:

- MNIST, handwritten digits, 28x28 greyscale images [17],
- Kuzushiji-MNIST, or K-MNIST, cursive Japanese characters, 28x28 greyscale images [18],
- Fashion MNIST, or F-MNIST, Zalando's article images, 28x28 greyscale images [19],
- CIFAR-10, 32x32 RGB images [20].

These datasets are organized into classes, such as handwritten digits in the MNIST dataset or types of vehicles and animals in CIFAR-10. We created hierarchical label structures by aggregating these classes into two coarse-grained, larger classes in different ways. For example, digits from 0 to 7 can be grouped into the two larger classes corresponding to even and odd numbers. All image pixels are normalized to the range [0, 1].

Different aggregations allow us to test how the relationship between fine and coarse label structures affects model performance. For CIFAR-10, a natural grouping separates animals [cats, dogs, deers, birds] from vehicles [cars, trucks, airplanes, ship]. Alternative fine-label organizations into coarse classes define different coarse tasks.

In addition to these real-world datasets, we designed a **synthetic dataset** to quantitatively control the geometric relationship between fine and coarse label boundaries. The synthetic data consists of points arranged in concentric circles with controllable boundary redundancy, allowing us to systematically test how data geometry affects the advantage of fine-grained training. The full construction and properties of this dataset are described in Section 2.4.3 of the Results, where its motivation and use are presented in context.

2.3.2 Model architectures and training

We compared two neural network architectures trained at different levels of label granularity. Both models are fully connected networks with a single hidden layer using ReLU activations.

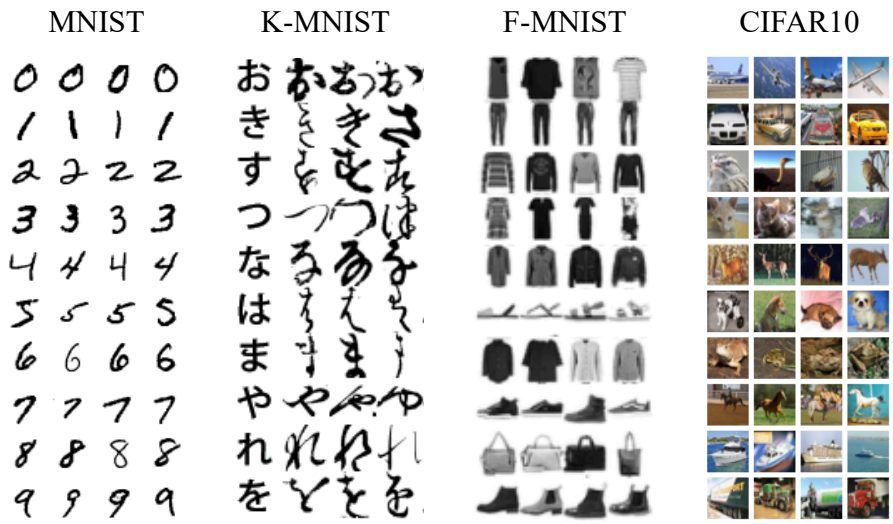

Figure 2.1: **Example images from the four datasets used in this study.** Each row shows samples from each of the ten classes. From left to right: MNIST (handwritten digits 0-9), K-MNIST (cursive Japanese characters), F-MNIST (clothing items: t-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, ankle boot), and CIFAR-10 (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck).

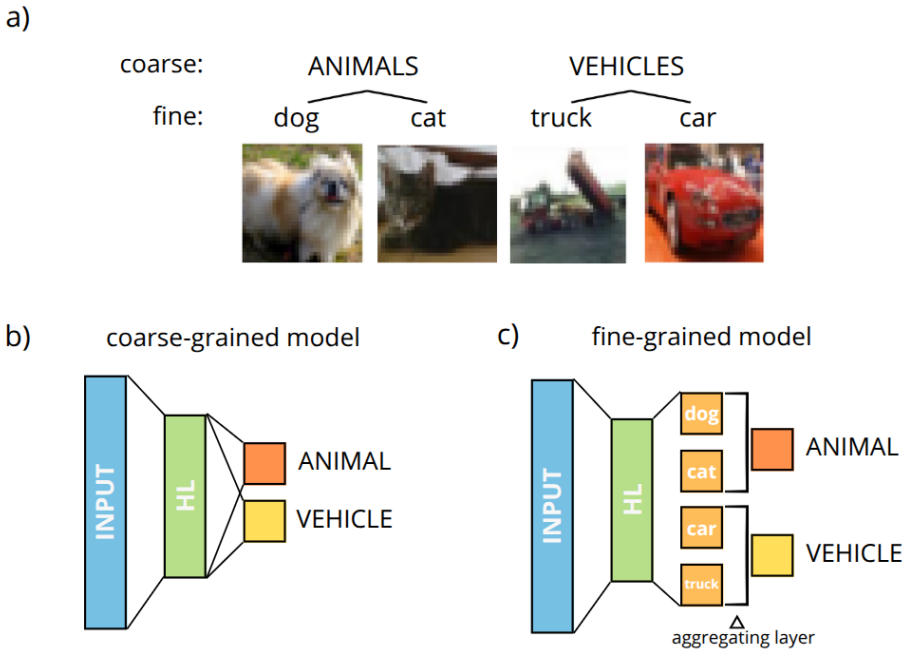

Figure 2.2: **Coarse and fine labeling.** (a) A possible way to categorize four CIFAR-10 classes (dogs, cats, cars, and trucks) into two broad coarse classes (animals and vehicles). Two models are employed during training: (b) coarse-grained and (c) fine-grained. The coarse-grained model is trained to differentiate between broad categories (e.g., animal vs. vehicle in the image). Instead, the fine-grained model is trained to classify specific subclasses within those categories. During testing, the fine-grained model is adapted to predict coarse classes by introducing an "aggregating layer" (shown in orange in the figure). This layer sums the output probabilities of the fine-grained model into binary predictions, aligning with the coarse class structure

The fine-grained model has K output neurons corresponding to the K fine subclasses. Given a dataset $\mathcal{D} = \{x^\mu, y^\mu\}_{\mu=1}^P$ where $x^\mu \in \mathbb{R}^d$ are feature vectors and $y^\mu \in \mathbb{R}^K$ are one-hot encoded fine labels, the fine model computes:

$$\hat{y}_i^\mu = \frac{\exp(z_i^\mu)}{\sum_{j=1}^K \exp(z_j^\mu)}, \quad z_i^\mu = \sum_{h=1}^{N_{\text{fine}}} w_{ih}\text{ReLU}(\omega_h \cdot x^\mu + b_h) + \beta_i, \tag{2.1}$$

where N_{fine} is the number of hidden neurons, $\omega_h \in \mathbb{R}^d$, b_h, w_{ih}, $\beta_i \in \mathbb{R}$ are learnable parameters. The softmax activation produces class probabilities $\hat{y}_i^\mu$ for each fine class i.

The coarse model is a binary classifier with a sigmoid output:

$$\hat{Y}_c^\mu = \sigma\left(\sum_{i=1}^{N_{\text{coarse}}} v_i\text{ReLU}(\omega_i \cdot x^\mu + b_i) + \beta\right), \tag{2.2}$$

where N_{coarse} is the number of hidden neurons, $\sigma(\cdot)$ is the sigmoid function, and $\omega_i \in \mathbb{R}^d$, b_i, v_i, $\beta \in \mathbb{R}$ are the model's parameters. The model is trained using binary cross-entropy loss on coarse labels $Y^\mu = \sum_{k\in\mathcal{C}_0} y_k^\mu$, where $\mathcal{C}_0$ and $\mathcal{C}_1$ denote the two coarse classes.

All model weights were initialized using Glorot uniform initialization and biases were set to zero. To ensure fair comparison, we tuned the number of hidden neurons N_{coarse} and N_{fine} so both architectures had approximately the same number of learnable parameters. The results do not depend on this precise architectural matching.

2.3.3 Adapting fine predictions to coarse labels

The fine model's prediction on the coarse task is obtained by summing the predicted probabilities of all subclasses belonging to each coarse category. An aggregating layer sums the softmax outputs over fine classes within the same coarse class:

$$\hat{Y}_f^\mu = \sum_{k\in\mathcal{C}_0} \hat{y}_k^\mu. \tag{2.3}$$

This coarse prediction $\hat{Y}_f^\mu$ represents the model's confidence that example μ belongs to coarse class $\mathcal{C}_0$.

Both models were evaluated on the coarse classification task. Test accuracy for each model is the proportion of correctly predicted coarse labels:

$$\text{Acc}_{\text{coarse}} = \frac{1}{P}\sum_{\mu=1}^P \mathbb{I}\left[\Theta\left(\hat{Y}_c^\mu - \frac{1}{2}\right) = Y^\mu\right], \quad \text{Acc}_{\text{fine}} = \frac{1}{P}\sum_{\mu=1}^P \mathbb{I}\left[\Theta\left(\hat{Y}_f^\mu - \frac{1}{2}\right) = Y^\mu\right], \tag{2.4}$$

where $\Theta(\cdot)$ is the Heaviside step function and $\mathbb{I}[\cdot]$ is the indicator function. Training accuracy was computed the same way on the training set.

Networks were trained using stochastic gradient descent with early stopping. The learning rate was linearly decreased from 0.01 to 0.001 during training. Batch size scaled with dataset size: 8 for the smallest datasets and 32 for the largest. Each experiment was repeated 30 times with different random weight initializations to assess variability.

For the synthetic circle dataset experiments with $K = 8$, we used different hyperparameters. The fine model had 60 hidden neurons, while the coarse model had 150 hidden neurons to equalize capacity. Both models were trained with Adam optimizer (learning rate 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$) with early stopping. The training set consisted of 5,000 points, and the test set consisted of 3,000 points.

2.4 Results

2.4.1 Fine-grained labels are not always optimal for training

The first question we investigate is whether training on fine-grained labels consistently improves generalization performance. Previous evidence, particularly from medical imaging and large-scale computer vision tasks, suggested that this approach is generally beneficial. To test this more systematically, we selected several standard datasets and organized their classes into different hierarchical structures. Specifically, we define two broad coarse classes, each composed of four fine-grained subclasses, and compare model performance on the coarse binary classification task when trained at the two levels of label granularity. For these experiments, we used the following coarse groupings: numbers 0 to 3 against numbers 4 to 8 for MNIST; [Coat, sandal, dress, ankle boot] vs. [pullover, sneaker, shirt, bag] for F-MNIST; [o, ki, su, tsu] vs. [na, ha, ma, ya] for K-MNIST; and vehicles [airplane, automobile, ship, truck] vs. animals [dog, deer, bird, cat] for CIFAR-10.

Figure 2.3a shows this comparison for two neural networks with the architectures described in Methods, trained on subsamples of the K-MNIST dataset of varying sizes. The model trained on fine-grained labels outperforms the coarse-grained one only for small dataset sizes. In contrast, training the network directly on coarse labels proves to be the better strategy for larger dataset sizes. This result challenges the conventional wisdom that fine-grained training is universally advantageous. The advantage of fine-grained supervision depends critically on the amount of available training data.

Training on fine-grained labels contributes to performance gains primarily by enhancing the model's ability to generalize. When the fine-grained model generalizes better, both models still interpolate nearly perfectly on the training set, as shown in

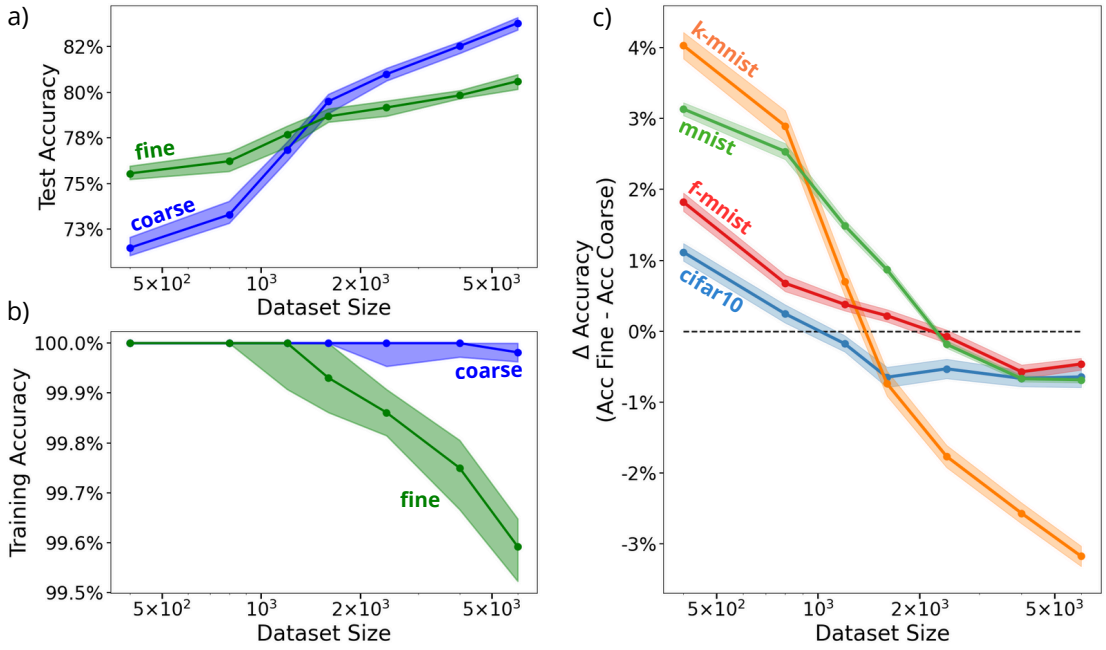

Figure 2.3: **The advantage of fine-grained training depends on dataset size.** (a) Test accuracy on K-MNIST as a function of training set size for neural networks trained on fine-grained labels (green) and on coarse-grained labels (blue). (b) Training accuracy for the same models. Both panels show networks with 10 hidden neurons. Shaded regions indicate the interquartile range (first to third quartile) over 30 independent training runs with different random initializations. (c) The difference in test accuracy, $\Delta\text{Accuracy} = \text{Acc}_{\text{fine}} - \text{Acc}_{\text{coarse}}$, across datasets (CIFAR-10, K-MNIST, F-MNIST, MNIST) at different dataset sizes. Networks have 10 hidden neurons for MNIST, F-MNIST, and K-MNIST, and 60 hidden neurons for CIFAR-10. Error bars represent the standard error over 30 runs.

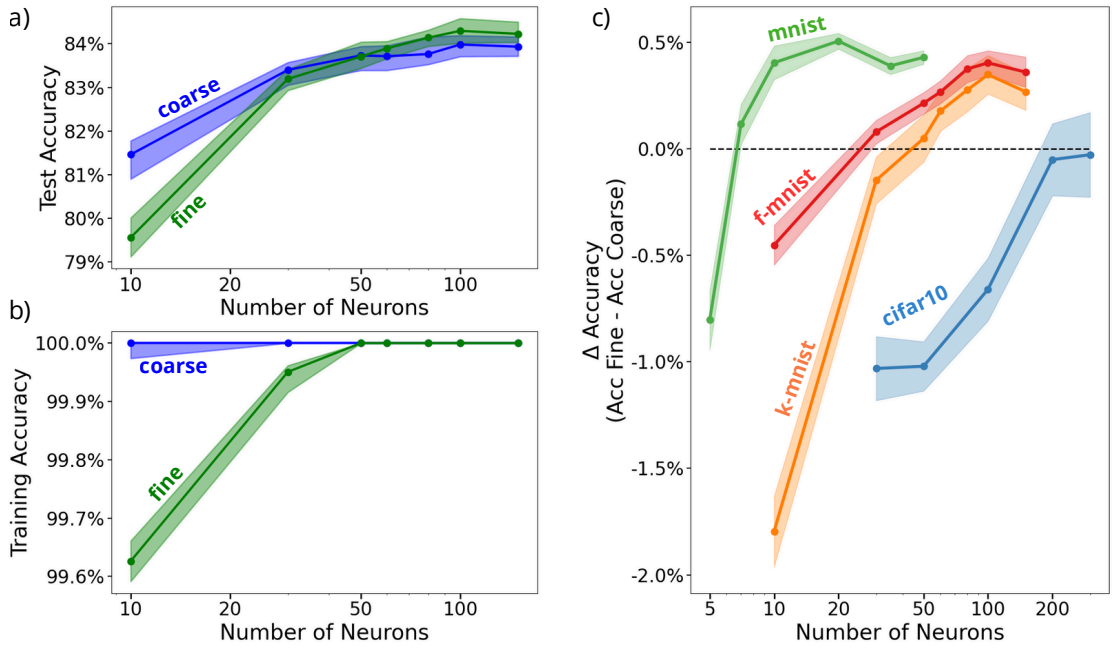

Figure 2.4: **The advantage of fine-grained training depends on model capacity.** (a) Test accuracy on K-MNIST as a function of the number of hidden neurons for neural networks trained on fine-grained labels (green) and on coarse-grained labels (blue), at fixed dataset size. (b) Training accuracy on the coarse task for the same models. Shaded regions in both panels indicate the interquartile range (first to third quartile) over 30 independent training runs with different random initializations. (c) The difference in test accuracy, $\Delta\text{Accuracy} = \text{Acc}_{\text{fine}} - \text{Acc}_{\text{coarse}}$, across all four datasets as a function of model capacity. Training set sizes are 1200 examples for MNIST and 3200 examples for F-MNIST, K-MNIST, and CIFAR-10. Error bars represent the standard error over 30 runs.

Figure 2.3b. However, as the dataset size increases and the fine-grained model begins to be outperformed, we observe a drop in its training accuracy when evaluated on the coarse labels. This suggests that, in certain settings, the more challenging task of distinguishing fine-grained classes can hinder optimization and, as a result, degrade final performance. This observation generalizes to other datasets: coarse-trained networks can match or exceed the training accuracy of their fine-grained counterparts, but tend to overfit more for small dataset sizes. This pattern is consistent across configurations, indicating that the optimization difficulty imposed by fine-grained training plays a role in its effectiveness.

The transition to a regime in which fine-grained training is no longer advantageous is consistently observed across all four real-world datasets we analyzed, as shown in Figure 2.3c.

Moreover, a similar trend emerges when comparing architectures with varying numbers of hidden neurons. Figure 2.4a shows test accuracy as a function of model capacity for K-MNIST, keeping the dataset size fixed. For networks with few hidden neurons,

the coarse-trained model outperforms the fine-grained one. As the number of neurons increases, the fine-grained model's performance improves more rapidly, eventually surpassing the coarse model. This transition mirrors what we observed when varying dataset size.

The mechanism behind this capacity-dependent transition is analogous to the dataset size case. Figure 2.4b shows that for high neuron counts, both models achieve near-perfect training accuracy on the coarse task. In this regime, the fine-grained model benefits from better generalization, similar to what we saw with small datasets. However, for networks with fewer neurons, the fine-grained model struggles to reach the same training accuracy as the coarse model. The additional complexity of discriminating between fine classes hinders optimization when model capacity is limited, preventing the network from fully solving even the coarse task.

Figure 2.4c demonstrates that this capacity-dependent transition is consistent across all four datasets. CIFAR-10 presents an interesting case: fine-grained training remains disadvantageous across the entire range of capacities tested, never crossing zero. However, the trend is analogous, with the performance gap narrowing as capacity increases. This behavior likely reflects the complexity of CIFAR-10 relative to our shallow network architecture. Even with hundreds of neurons, these single-hidden-layer networks struggle to fully fit the training set, as evidenced by the training accuracies remaining below saturation. The dataset is simply too complex for shallow networks to capture effectively, regardless of width. This suggests that the transition we observe in simpler datasets (MNIST, F-MNIST, K-MNIST) requires not just sufficient capacity, but also the ability to actually solve the training task, which CIFAR-10's complexity prevents in our architectural setting.

These two observations, that dataset size and network capacity both influence the relative performance of fine versus coarse training, can be unified by considering the degree of model overparameterization. Whether a model is overparameterized depends on the interplay between these two quantities: a small network can still be in the overparameterized regime if the dataset is small, while a large network may be underparameterized if faced with abundant data. The parameter-to-data ratio n/p, where n is the total number of learnable parameters and p is the dataset size, naturally captures this interplay. Figure 2.5a shows the advantage of training on fine-grained labels as a function of n/p across all the experiments for all datasets. Each point represents a specific combination of dataset size and model capacity. We observe a clear trend: fine-grained models tend to perform better at higher levels of overparameterization, corresponding to larger n/p values. This unified view confirms that neither dataset size nor model capacity alone determines the outcome. Rather, it is their ratio that matters, with fine-grained training becoming more beneficial as we move toward more overparameterized regimes.

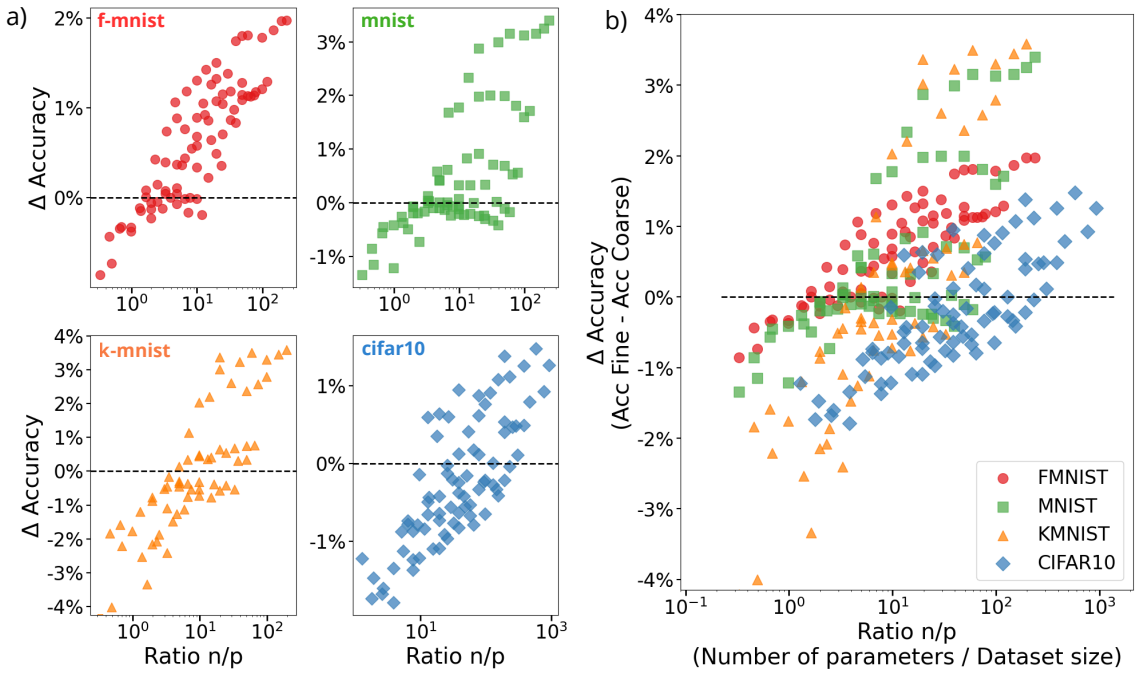

Figure 2.5: **The advantage of fine-grained training unifies under the parameter-to-data ratio.** (a) $\Delta\text{Accuracy} = \text{Acc}_{\text{fine}} - \text{Acc}_{\text{coarse}}$ plotted against the ratio of model parameters to dataset size (n/p) for each dataset separately. (b) All datasets combined, showing the general trend across different data types. Each point represents a specific combination of dataset size and model capacity, averaged over 30 independent training runs. Hidden layer sizes span 5, 7, 10, 20, 25, 30, 50, 75, 100, and 120 neurons. Training set sizes span 400, 800, 1200, 1600, 2400, 4000, 8000, and 12000 examples. The trend shows that fine-grained training becomes more beneficial at higher overparameterization ratios, though with dataset-specific variability visible in panel (a).

However, as displayed in Figure 2.5b, the transition point appears to be dataset-specific, and the large fluctuations suggest the presence of additional relevant control variables. The variability across datasets indicates that factors beyond simple overparameterization matter.

2.4.2 The role of the label hierarchical structure, data geometry, and the fine-coarse task alignment

Training on fine-grained labels does not always boost model performance, and seems generally helpful when data is scarce compared to the number of parameters. However, we focused on a single coarse task, while the categories of a given dataset can be aggregated into different coarse classes, depending on the task. Consider, for example, a dataset with fine classes like sandals, boots, t-shirts, and coats. One could define a coarse task separating shoes from clothes, but equally valid would be grouping winter items against summer items. These alternative organizations generate different coarse

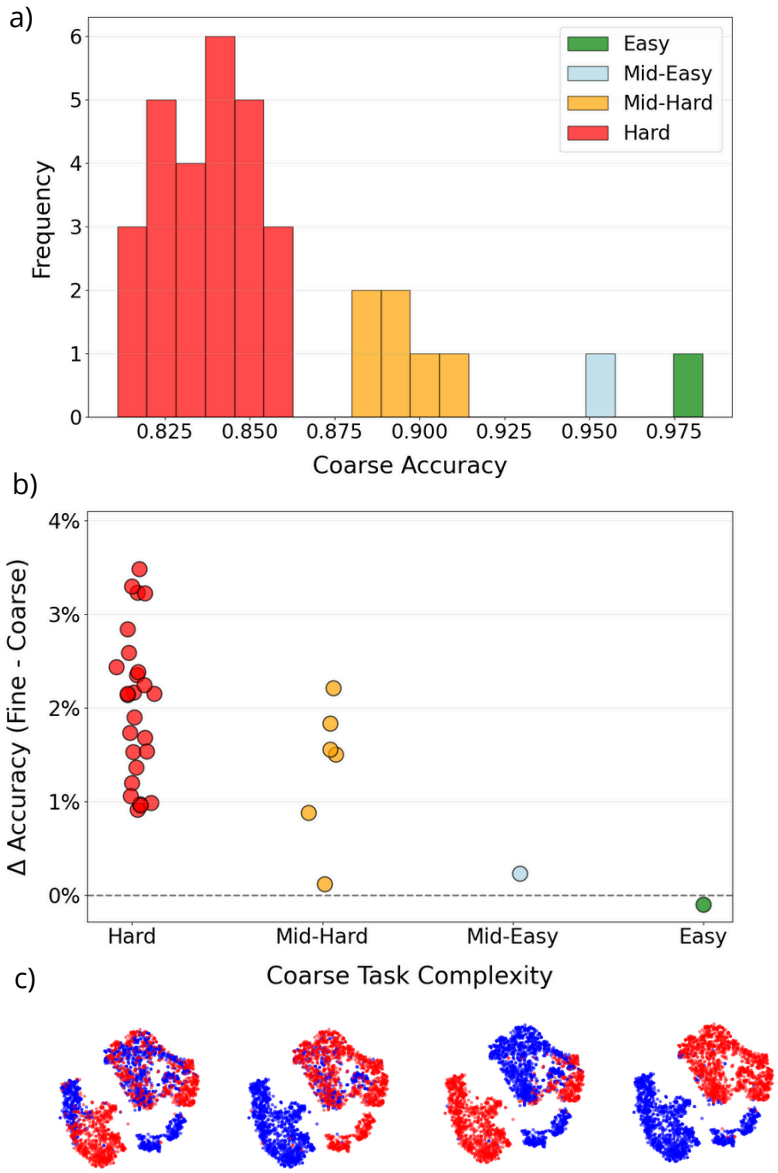

Figure 2.6: **Systematic exploration of coarse task difficulty in Fashion-MNIST**. (a) Distribution of coarse accuracies across all 35 possible ways to partition 8 fine classes into two balanced coarse classes of 4 classes each. Every accuracy value averaged over 30 independent runs. Colors indicate qualitatively identified difficulty clusters based on the histogram: easy (green), mid-easy (light blue), mid-hard (orange), and hard (red). (b) Performance gain from fine-grained training ($\Delta\text{Accuracy} = \text{Acc}_{\text{fine}} - \text{Acc}_{\text{coarse}}$) as a function of task complexity. Each point represents one of the 35 configurations and is the result of an average over 30 independent runs. Fine-grained training provides substantial benefits for hard tasks (up to 3.5%) but no advantage for easy tasks. Both models were trained with 400 examples per coarse class using identical architectures. (c) t-SNE visualizations of one representative configuration from each difficulty cluster, applied directly to raw data points. Colors represent the two coarse classes. Hard tasks show substantial mixing of coarse classes in the embedding space, while the easy task shows cleaner separation.

tasks with different levels of complexity, and potentially different relationships between label structure and the geometry of the data. Understanding whether such arrangements systematically affect the utility of fine-grained training could reveal something general about how label organization interacts with the underlying structure of visual data.

To explore this question, we used Fashion-MNIST, which contains 8 classes that can be partitioned into two balanced coarse tasks in 35 different ways. For each configuration, we trained two neural networks with identical architectures: one on the fine-grained 8-class task and one on the corresponding coarse 2-class task. We then measured the coarse accuracy of both as explained in the Methods section. Since the architecture and dataset size were held constant across all experiments, differences in coarse accuracy reflect the intrinsic difficulty of each coarse task.

Figure 2.6 shows how these 35 tasks distribute by difficulty. Panel a) presents an histogram of coarse accuracies, revealing that most configurations yield relatively hard tasks, with accuracies around 82–87%, while only a few result in easier tasks with accuracies above 90%. By inspecting the histogram, we qualitatively identified four *difficulty clusters*. These difficulty levels align with semantic intuition: the easiest configuration separates clothing items from shoes and bags, which represent distinct functional categories. In contrast, the hardest tasks mix semantically similar items across both coarse classes, such as grouping sandals, dresses, bags, and shirts together while placing ankle boots, coats, sneakers, and pullovers in the other class. Moving from easy to hard configurations involves swapping individual classes. Intermediate difficulty levels are often generated by swapping semantic outliers, i.e. bags. Because bags do not possess the visual features of either clothing or footwear, shifting them creates middle-ground configurations that blur the semantic boundaries without causing the total class confusion seen in the hardest tasks.

Having established these difficulty levels, we examined whether the benefit of fine-grained training varied systematically across them. For this analysis, we used 400 training examples per coarse class, a dataset size that places us in the low-data regime where, based on the previous section's findings, fine-grained training should generally be beneficial. Panel b) plots the performance gain $\Delta\text{Accuracy} = \text{Acc}_{\text{fine}} - \text{Acc}_{\text{coarse}}$ against task complexity for all 35 configurations. A clear pattern emerges: fine-grained training provides substantial benefits for hard tasks, with improvements reaching up to 3.5%, but offers minimal or no advantage for easy tasks. This variation is striking precisely because we are in a regime where fine training was expected to help universally. We focus on this low-data regime to highlight the dependence of fine-advantage from task structure, but the full accuracy curves as a function of dataset size, for representative configurations from each complexity cluster, are shown in Appendix A.1.

This relationship between task difficulty and the utility of fine labels suggests a

geometric explanation. To explore this intuition, we visualized one example from each difficulty cluster using t-SNE applied directly to the raw image pixels. Figure 2.6c shows these visualizations. While t-SNE is purely qualitative and may introduce artifacts, the patterns provide a suggestive hint. For hard tasks, the two coarse classes appear geometrically intricate, with substantial mixing between fine subclasses from different coarse categories. For the easy task, the coarse classes show cleaner separation in the embedding space.

This systematic analysis across Fashion-MNIST revealed a pattern: the benefit of fine-grained training appears to depend on how the discriminative features needed to separate fine classes relate to those needed for the coarse task. However, the semantic complexity of Fashion-MNIST categories and the qualitative nature of these observations made it difficult to isolate the key geometric factors at play. To demonstrate this principle more clearly, we designed a controlled experiment using CIFAR-10, constructing coarse tasks with specific geometric properties.

For a dataset like CIFAR-10, there is an inherent semantic organization of labels. A natural choice would be to separate animals, divided into the fine classes of cats and dogs, from vehicles, divided into cars and trucks. However, different coarse tasks could be defined by alternative label organizations. For instance, one could compose a coarse class with cats and trucks, and another with dogs and cars, as illustrated in Figure 2.7a.

We hypothesize that training on fine labels adds information about the coarse classes only when the discriminative features necessary to separate fine subclasses are also relevant for the coarse task. If this is the case, the specific class organization should be crucial in defining the amount of this additional information and its relevance for the task. To formalize this intuition, we introduce the concept of boundary redundancy, which captures the level of alignment between the tasks defined by coarse and fine labels.

Consider a coarse classification task, such as separating animals from vehicles. In this case, some features necessary to solve the coarse task, such as the presence of wheels or paws, may allow the model to solve the coarse task perfectly without necessarily learning to differentiate between fine-grained subclasses, such as cats and dogs or trucks and cars. Instead, a model trained on fine-grained labels must learn distinctions that are not necessarily relevant for the coarse task. In other words, the boundaries in the data space that the fine-trained model has to discover are redundant with respect to the coarse task on which performance is evaluated. This is the case illustrated on the left of Figure 2.7a and b.

In contrast, coarse classes could be composed of more granular categories whose discriminative features coincide with the ones needed for the coarse task, as shown in the right panel of Figure 2.7a. From a geometric point of view, the boundaries in the

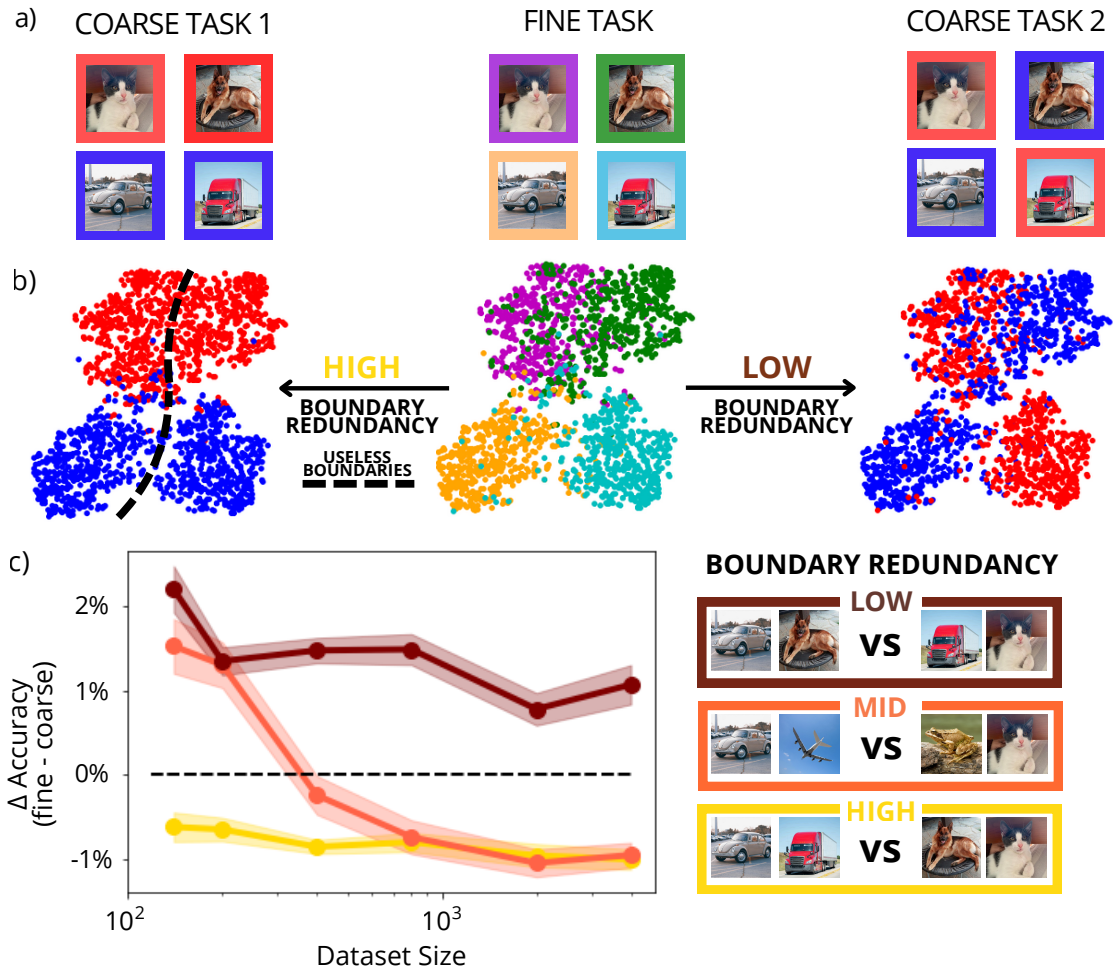

Figure 2.7: **The impact of boundary redundancy on generalization.** (a) Four CIFAR-10 classes (*cat*, *dog*, *car*, *truck*) are grouped into two alternative coarse classification tasks. Coarse Task 1: animals vs. vehicles, where subclasses share natural attributes. Coarse Task 2: (*cat*, *truck*) vs. (*dog*, *car*), which mixes dissimilar subclasses. (b) t-SNE embedding of the internal representation learned by a convolutional neural network trained on the fine-grained four-class task (see details in App. A.1). Points are recolored according to the coarse tasks: red vs. blue for the two coarse classes. In this space, some fine-grained boundaries are superfluous for Coarse Task 1 but remain necessary for Coarse Task 2. (c) Relative performance of the fine-trained vs. coarse-trained models for three coarse tasks and different dataset sizes. The legend indicates the supposed level of boundary redundancy for each task. Models are one-hidden-layer networks with 30 hidden neurons, trained with SGD and early stopping. Averages are taken over 30 runs with different random initializations. The test set consists of 10^4 points, and the error bars represent the standard error.

data space resulting from the two optimization strategies largely overlap, meaning that redundancy is low.

Therefore, for class organizations with high boundary redundancy, training on fine-grained labels may burden the optimization process with distinctions that are irrelevant to the task, potentially leading to equal or even worse performance. In contrast, for class structures with low boundary redundancy, we hypothesize that fine-grained training is more likely to outperform its coarse counterpart, as the learning process focuses on the relevant decision boundaries while benefiting from additional structural information.

Figure 2.7a provides three illustrative examples of class organizations with high, medium, and low boundary redundancy, constructed using the CIFAR-10 dataset. A two-dimensional t-SNE embedding of the data supports the geometric intuition behind the different levels of boundary redundancy, as shown in Figure 2.7b. To produce this visualization, we trained a convolutional neural network on the four-class fine-grained task and extracted the internal representations from the last hidden layer. These high-dimensional representations were then projected into two dimensions using t-SNE (details in App. A.1). While t-SNE may introduce visualization artifacts and does not preserve exact geometric relationships, the qualitative patterns provide intuitive support for our concept of boundary redundancy. In the low redundancy case, on the right panel, the two coarse classes are well separated in the representation space, with minimal mixing between fine subclasses belonging to different coarse categories. In the high redundancy case, on the left panel, fine subclasses from different coarse classes overlap substantially in the representation space, indicating that the features learned to separate fine classes do not align well with the coarse classification boundary.

To test our hypothesis, we trained two neural networks on fine and coarse-grained groupings with different levels of boundary redundancy, and compared their performance at different dataset sizes. The results are shown in Figure 2.7c, where we consider the gain in performance $\Delta\text{Accuracy} = \text{Acc}_{\text{fine}} - \text{Acc}_{\text{coarse}}$ that training on fine-grained labels can provide relative to directly training the model on coarse labels. For a task expected to have high boundary redundancy, the fine-grained model indeed never outperforms the coarse-grained one. In contrast, the fine-grained model consistently generalizes better on a task with low boundary redundancy. At an intermediate level of boundary redundancy, we observe the transition described in the previous section: the fine-trained model performs better for smaller dataset sizes but is eventually outperformed as more data becomes available.

Although this result fully supports our hypothesis, the notion of boundary redundancy is based mainly on intuition and is difficult to quantify precisely. The t-SNE visualizations provide qualitative support, but we cannot measure redundancy in a controlled, quantitative manner using real-world datasets. Therefore, the next section

introduces a synthetic dataset in which redundancy can be measured and controlled, allowing a more quantitative test of the relationship between data geometry and the advantage of fine-grained training.

2.4.3 Testing the importance of boundary redundancy using a controlled synthetic dataset

To quantitatively control the level of boundary redundancy inherent in a label structure, we introduce a synthetic dataset based on a radial function. These kinds of functions are known to be difficult for shallow neural networks to learn [21], making them a suitable choice for constructing a sufficiently complex task. At the same time, their two-dimensional nature allows for straightforward visualization and interpretation. By adjusting the geometry of the data points across subclasses, we can continuously tune the degree of redundancy.

More specifically, the two-dimensional dataset consists of data points (x_1, x_2) arranged in K concentric circles. The two coarse labels are assigned to points on a given circle alternating between class A and class B as the circle radius increases, as illustrated in Figure 2.8a. Thus, the coarse class of a point $x^\mu = (x_1^\mu, x_2^\mu)$ is given by

$$Y^\mu = \begin{cases} 0 & \text{if } \|x^\mu\| = \frac{2i}{K}, \\ 1 & \text{if } \|x^\mu\| = \frac{2(2i-1)}{K}, \end{cases} \qquad i = 1, \dots, K/2, \tag{2.5}$$

where the label 0 corresponds to class A, and the label 1 to class B.

Each one of these coarse classes is then partitioned into $K/2$ subclasses, as shown in Figure 2.8b, and we engineered this partitioning to create structures with different levels of redundancy. The simplest way to define the subclasses is to assign each circle to a distinct subclass, as illustrated in Figure 2.8c:

$$y^\mu = j \quad \text{if } \|x^\mu\| = r_j, \quad j = 1, \dots, K, \tag{2.6}$$

where r_j is the radius of the j-th circle. This configuration represents the structure with the lowest level of boundary redundancy. To separate the fine classes, meaning individual circles, the model has to identify the same boundaries needed to separate the two coarse classes.

To tune the level of boundary redundancy, we can redefine the subclasses by mixing data regions belonging to different circles, as shown in Figure 2.8d and e. The level of mixing can be quantified by measuring the fraction of each circle that is occupied by a subclass different from the predominant one. This fraction determines the extent to which the task defined by fine and coarse training overlaps. When each circle defines a distinct subclass, the boundary redundancy is zero, as in Figure2.8c. Figure 2.8d

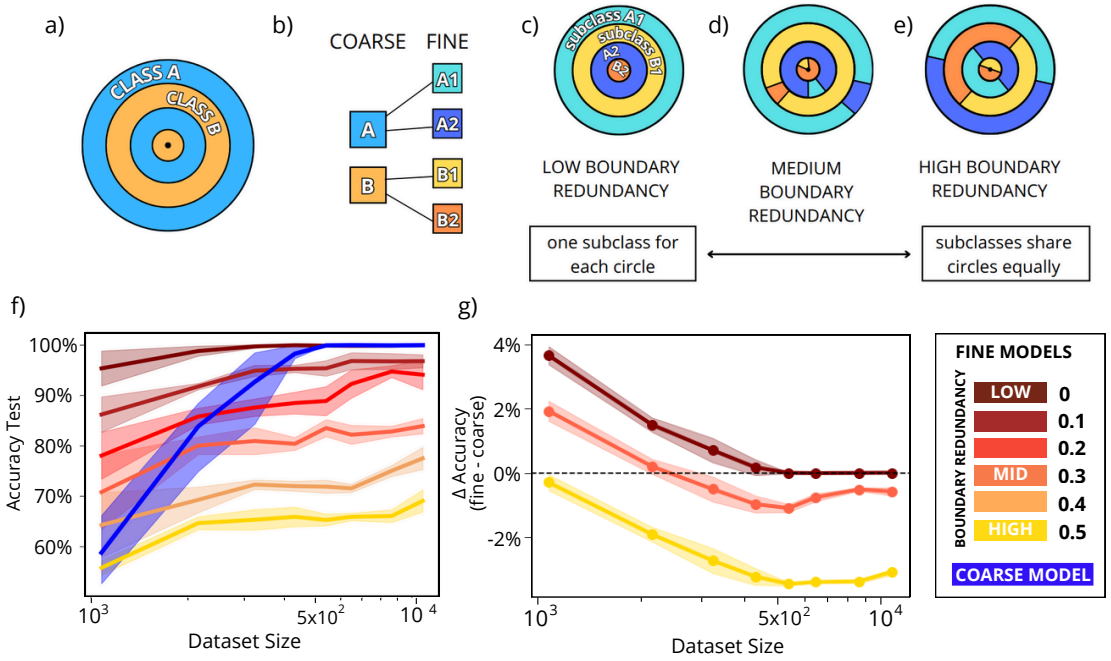

Figure 2.8: **Experiments with synthetic data confirm the role of data geometry for the advantage of fine-grained training.**(a) Dataset construction: four concentric circles ($K = 4$), with randomly distributed points. Coarse labels alternate between class A (blue) and class B (orange) from one circle to the next. (b) Hierarchical label structure obtained by splitting the two coarse classes into two subclasses. (c-e) The geometry of the fine classes can define problems with different boundary redundancy. (c) No redundancy: each circle only contains elements of one subclass, and the fine-grained task and the coarse one coincide. (d,e) By changing the fraction of points in each circle belonging to another subclass, the fine task diverges from the coarse one. Highest redundancy is obtained when subclasses are equally represented in each circle. (f-g) Performance of fine- vs. coarse-grained models trained on the synthetic circle dataset with $K = 8$ as a function of dataset size (N), and different values of BR. Models trained with fine subclass structures of varying BR are colored in different shades of red, while the blue line represents the model trained on coarse labels. To equalize capacity, both models were matched for trainable parameters: the fine model uses 60 hidden neurons, the coarse model 150. Both models were trained with Adam (lr= 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$), with early stopping. The training set consists of 5,000 points, and the test set consists of 3,000 points. In (f), the error bars indicate the range between the first and third quartiles of the data obtained on 30 runs with different initialization, while in (g), the error bars are the expected standard error.

shows an intermediate case in which each circle contains 70% of points assigned to a dominant subclass and the remaining 30% to another subclass, thus corresponding to a redundancy level of 0.3. The highest possible redundancy is $1 - 2/K$, as represented in Figure 2.8e. In this maximal redundancy configuration, all subclasses are equally represented in each circle, meaning that distinguishing fine classes provides almost no useful information for the coarse task. All configurations share the same coarse label structure: even when a single circle contains points from multiple subclasses, all points on that circle still belong to the same coarse class, following the hierarchical organization illustrated in Figure 2.8b.

To connect this synthetic construction to our earlier examples, the zero redundancy case in Figure 2.8c is analogous to the low redundancy CIFAR-10 configuration from Figure 2.7 (right panel), where cats and trucks form one class and dogs and cars form another. In both cases, learning to separate the fine subclasses requires identifying the same boundaries needed for the coarse task. Conversely, the maximum redundancy configuration in Figure 2.8e mirrors the high redundancy case from Figure 2.7 (left panel), where animals are grouped separately from vehicles. Here, the model is forced to learn distinctions within each coarse class, such as separating individual circles into multiple subclasses, even though the coarse task only requires distinguishing between circles.

Figure 2.8f shows the test accuracy on the coarse classification task for models trained on fine-grained labels with varying levels of boundary redundancy. As reported in the previous sections, fine-trained models with a fixed architecture outperform the coarse-trained model, shown as the blue curve, only for small dataset sizes. As we hypothesized, the accuracy achieved by fine-trained models scales with the boundary redundancy of the label structure. Models trained on label structures with low redundancy, shown in brown, achieve the highest accuracy and maintain their advantage over coarse training even for relatively large datasets. As redundancy increases, the performance of fine-trained models decreases, and the advantage over coarse training diminishes more rapidly as dataset size grows.

From the results in previous section we expect that, as the dataset size increases, the benefit of fine-grained supervision diminishes, as shown in Figure 2.8g. In addition, here we see that only label structures with low boundary redundancy continue to offer an advantage in training for larger datasets. The transition point, meaning the dataset size at which training on fine labels is preferable, crucially depends on the data geometry, in particular on the boundary redundancy. When there is no boundary redundancy, shown by the brown curve, the problem of separating the fine classes is geometrically equivalent to the coarse class discrimination. In this case, the fine-grained training provides explicit supervision on the same boundaries needed for the coarse task, and the model consistently benefits from this additional information

across all dataset sizes tested. As redundancy increases, the curves shift downward, indicating that fine-grained training becomes less beneficial and eventually harmful. At intermediate redundancy levels, we observe a crossover where fine training helps for small datasets but hurts for large ones. This is what we normally encounter in real datasets, where there is neither perfect separation nor complete mixing of relevant features across the label hierarchy. In fact, this is exactly the behavior we observed with real datasets in the first section. At the highest redundancy level, shown in light red, fine training never provides a benefit, consistent with the intuition that forcing the model to learn irrelevant distinctions burdens optimization without improving coarse-level performance.

These synthetic experiments confirm that boundary redundancy is a key factor determining when fine-grained training helps. The ability to continuously vary redundancy in a controlled setting allows us to establish a quantitative relationship between geometric alignment and training effectiveness. This validates our hypothesis from the previous section and provides a concrete mechanism explaining why the transition point between beneficial and detrimental fine-grained training varies across real-world datasets.ù

2.4.4 Insights on the relation between fine and coarse training from a loss function decomposition

The previous sections demonstrated empirically that the benefit of fine-grained training depends on the geometric alignment between fine and coarse decision boundaries. To understand the mechanism behind this observation, we can examine what happens at the level of the loss function. By decomposing the fine-grained cross-entropy loss, we can formalize how training on fine labels affects the optimization of the coarse task, and why boundary redundancy plays such a crucial role.

We consider a setting in which there are K fine-grained classes, denoted by the labels $y_i \in \{0, 1\}$, with $i = 1, \ldots, K$. Each data point belongs to exactly one fine class, so that $\sum_{i=1}^{K} y_i^\mu = 1$, where the index μ identifies the data point.

We introduce two coarse disjoint classes with index sets $\mathcal{C}_0$ and $\mathcal{C}_1$, such that $\mathcal{C}_0 \cup \mathcal{C}_1 = \{1, \ldots, K\}$ and $\mathcal{C}_0 \cap \mathcal{C}_1 = \emptyset$. The corresponding coarse labels $Y^\mu \in \{0, 1\}$ are defined as $Y^\mu = \sum_{i \in \mathcal{C}_0} y_i^\mu$.

The fine-trained model outputs a probability distribution $\hat{y}_i^\mu$ over the fine classes using a softmax function. The predictions over the coarse classes $\hat{Y}^\mu$ are simply obtained by summation, given the known label structure, that is, $\hat{Y}^\mu = \sum_{i \in \mathcal{C}_0} \hat{y}_i^\mu$.

We can then formulate the loss for coarse training as a classic binary cross-entropy

$$\mathcal{L}_{\text{coarse}} = -\frac{1}{p}\sum_{\mu=1}^{p}\Big(Y^{\mu}\log(\hat{Y}^{\mu}) + (1-Y^{\mu})\log(1-\hat{Y}^{\mu})\Big) =$$

$$= -\frac{1}{p}\sum_{\mu=1}^{p}\left[\Big(\sum_{i\in\mathcal{C}_0} y_i^{\mu}\Big)\log\Big(\sum_{i\in\mathcal{C}_0}\hat{y}_i^{\mu}\Big) + \Big(\sum_{i\in\mathcal{C}_1} y_i^{\mu}\Big)\log\Big(\sum_{i\in\mathcal{C}_1}\hat{y}_i^{\mu}\Big)\right].$$

Analogously, the loss $\mathcal{L}_{\text{fine}}$ that is minimized during fine training is a cross-entropy loss over the K classes, defined as

$$\mathcal{L}_{\text{fine}} = \frac{1}{p}\sum_{\mu=1}^{p}\sum_{i\in\mathcal{C}_0\cup\mathcal{C}_1} y_i^{\mu}\log(\hat{y}_i^{\mu}).$$

However, this expression can be reformulated as

$$\mathcal{L}_{\text{fine}} = \mathcal{L}_{\text{coarse}} + \mathcal{L}_{\text{intra-class}},$$

where the intra-class loss is

$$\mathcal{L}_{\text{intra-class}} = \frac{1}{p}\sum_{\mu=1}^{p}\left[\sum_{i\in\mathcal{C}_0} y_i^{\mu}\log\Big(1+\frac{\sum_{j\in\mathcal{C}_0\setminus\{i\}}\hat{y}_j^{\mu}}{\hat{y}_i^{\mu}}\Big) + \sum_{i\in\mathcal{C}_1} y_i^{\mu}\log\Big(1+\frac{\sum_{j\in\mathcal{C}_1\setminus\{i\}}\hat{y}_j^{\mu}}{\hat{y}_i^{\mu}}\Big)\right].$$

This decomposition shows that training on fine-grained classes minimizes the coarse loss together with a term that depends on intra-class errors, namely, misclassifications of fine labels within a coarse class. If the boundaries that optimize the coarse and fine tasks largely coincide, minimizing the intra-class loss does not add substantial complexity to the optimization. Therefore, in this case, fine-training can improve predictions on the coarse-grained task. This is particularly true when the training set is small, as the additional term $\mathcal{L}_{\text{intra-class}}$ can effectively act as a regularizer, pushing the model - among all solutions that minimize the coarse loss - towards one that also separates subclasses reasonably well. This, in turn, can encourage better representations and improve generalization.

Conversely, when the boundaries for the two tasks differ substantially, minimization with the additional intra-class loss term can select solutions that make more errors on the coarse classes. This effect is particularly relevant for large training sets relative to the number of free parameters, for which the optimization is already strongly constrained.

2.5 Discussion

Hierarchical structures in data and learning

This work represents a first step toward understanding the role of hierarchical class structures in neural network training. Such structures appear everywhere. Standard categorical datasets are organized hierarchically, including classic computer vision benchmarks like ImageNet, which derives its semantic organization from WordNet, and CIFAR-100, where fine classes like "crocodile" and "lizard" belong to coarser groups like "reptiles." But hierarchical label structures are not limited to specific datasets. They reflect a general tendency to record and organize knowledge in progressively more detailed categories. The organization of Wikipedia pages follows hierarchical principles [22]. Words in natural language are arranged into hierarchies through structural and semantic relations, as exemplified by generative grammars [23] and lexical databases like WordNet [24]. Empirical studies have observed that we can use these hierarchical structures in machine learning, and that training on labels more fine-grained than the target task can improve performance [5, 6, 7, 8]. Our results show that the picture is more complex, revealing an interplay between data geometry, label organization, and model overparameterization.

Summary of findings

The experiments across multiple datasets and conditions establish three key results. First, fine-grained training does not universally improve coarse-level performance. The benefit depends critically on dataset size and model capacity: when training data is scarce relative to the number of model parameters, fine-grained labels tend to help; as data becomes abundant, this advantage disappears or reverses. Second, the geometric relationship between fine and coarse classification boundaries, which we call boundary redundancy, is central to understanding when fine-grained training succeeds. When fine-grained distinctions align with the coarse task, training on fine labels provides useful supervision on exactly the boundaries that matter; when they do not, it forces the model to learn distinctions irrelevant to the coarse objective. Third, task difficulty interacts with these effects in ways that were not previously characterized. In Fashion-MNIST, we found that hard coarse tasks, where the underlying data geometry makes classification intrinsically difficult, benefit substantially from fine-grained training, with improvements reaching up to 3.5%, while easy tasks show minimal gains regardless of the training strategy.

The role of the intra-class loss

The loss decomposition introduced in Section 2.4.4 provides a mathematical framework for understanding these phenomena. Training on fine-grained labels minimizes not only the coarse classification loss but also an additional term, the intra-class loss, that depends on errors in distinguishing fine classes within each coarse category. When the boundaries needed for fine and coarse classification largely coincide, minimizing this additional term does not substantially complicate the optimization. In this regime, the intra-class loss acts as a regularizer: among all solutions that minimize the coarse loss, it selects those that also separate subclasses reasonably well. This encourages better internal representations and improves generalization.

The regularizing effect is most pronounced when training data is limited relative to model capacity. In this regime, many different parameter configurations achieve similar coarse loss values, and the intra-class term provides additional guidance that breaks this degeneracy. For small datasets, this additional constraint helps prevent overfitting by pushing the model toward solutions with more structured internal representations. This explains why we consistently observe benefits of fine-grained training in the low-data regime across all datasets. However, when the boundaries required for fine and coarse tasks differ substantially, the intra-class loss becomes problematic. Minimization with this additional term can select solutions that make more errors on the coarse classes, because the optimizer is pulled toward boundaries that are geometrically incompatible with the coarse objective. This effect is most pronounced for large training sets relative to the number of free parameters. When data is abundant, the optimization is already strongly constrained by the coarse loss alone, and the additional fine-grained constraint can only interfere with the optimal solution.

Connection to previous work

Our findings help reconcile apparently conflicting reports in the literature. Chen et al. [6] found consistent benefits from fine-grained training across CIFAR-10, CIFAR-100, and ImageNet subsets. However, their experiments focused on overparameterized networks where the base coarse task was already nearly solved, with accuracies reaching 90–99%. In this regime, networks have abundant capacity to accommodate the additional complexity of fine labels without sacrificing coarse performance. Our results show that as we move away from this overparameterized setting, the benefit of fine-grained training becomes conditional on geometric factors. Esteva et al. [5] achieved remarkable success using fine-grained training for skin cancer diagnosis, with networks trained on 757 disease subtypes outperforming those trained directly on the three target coarse classes. The hierarchical organization of dermatological conditions likely exhibits low boundary redundancy: the features distinguishing malignant from benign

lesions are related to, though more general than, the features distinguishing specific malignant subtypes from one another. In such settings, our framework predicts that fine-grained training should help, which aligns with their clinical results. Hinton et al.'s work on knowledge distillation [14] provides a complementary perspective. They showed that soft targets from fine-grained class probabilities contain "dark knowledge" about similarity structures in the data. Our loss decomposition makes this intuition more concrete: the intra-class loss term explicitly quantifies the additional information carried by fine-grained labels. But our results add that this information is not always useful. Its value depends on whether the similarities encoded in the fine class structure align with the discriminative features needed for the coarse task.

Practical implications

For practitioners facing the choice between coarse and fine-grained training, our results suggest several guidelines. When data is limited, training on available fine-grained labels is likely to help, particularly if the fine classes reflect natural semantic or perceptual groupings within each coarse category. Medical imaging applications, where fine disease subtypes tend to share underlying pathological features with their coarser diagnostic categories, are well-suited to this approach. When data is abundant and the relationship between fine and coarse labels is unclear, the safer strategy may be to train directly on the target task. The additional complexity of fine-grained optimization carries risks when the geometric alignment is poor. The Fashion-MNIST experiments suggest another consideration: task difficulty. Easy coarse tasks, where the classes are already well-separated in feature space, gain little from fine-grained supervision regardless of other factors. The benefits of fine-grained training concentrate on hard tasks, where the model needs all the guidance it can get to find good decision boundaries.

Limitations

This study has several limitations that bound the generality of our conclusions. We deliberately focused on simple model architectures, specifically fully connected networks with a single hidden layer, and standard optimization techniques. This minimalistic approach, inspired by statistical physics [16], allowed for extensive quantitative exploration and transparent interpretation. However, the specific transition points and effect sizes we report may differ for deeper networks, convolutional architectures, or transformer-based models. A natural next step is to test whether our qualitative findings hold across a wider range of architectures. Our experiments also focused on binary coarse tasks, where fine classes are aggregated into exactly two coarse categories. Real hierarchical classification problems often involve more than two levels of granularity and

more than two coarse classes at each level. The intra-class loss decomposition generalizes naturally to the multi-class setting, but the geometric relationships become more complex. Whether boundary redundancy remains the key factor in these richer settings is an open question. The synthetic circle dataset provided controlled conditions for testing our hypotheses about boundary redundancy, but two-dimensional radial tasks are far from the complexity of real image classification. The connection between our quantitative redundancy measure and geometric properties of high-dimensional image data remains qualitative. Developing metrics for boundary redundancy that can be applied directly to real datasets would strengthen the practical applicability of our framework.

Future directions

Several extensions of this work could prove valuable. The loss decomposition suggests an immediate modification to explore: a training objective of the form $\mathcal{L} = \mathcal{L}_{\text{coarse}} + \beta\mathcal{L}_{\text{intra-class}}$, with $\beta \leq 1$ as a tunable hyperparameter. This would allow continuous interpolation between pure coarse training ($\beta = 0$) and pure fine training ($\beta = 1$), potentially offering better control over the trade-off between additional information and optimization complexity.

The presence of hierarchical label structures often reflects underlying compositionality in the features defining classes: general features distinguish coarse categories, while increasingly specific features resolve finer distinctions. Compositionality is widely regarded as central to neural network generalization [25, 26]. Our results suggest that understanding the specific relationship between label structure and data geometry could help design training strategies that use compositionality more effectively. Data can also be inherently compositional and organized hierarchically even when the attached labels do not reflect this organization. Hidden stratification can be inferred from the data and used to benefit training [13]. Our approach could therefore extend to unsupervised or semi-supervised settings, where hierarchical structure is discovered rather than provided.

Previous attempts to incorporate hierarchical class structures directly into neural network architectures, through hierarchical loss functions or dedicated classification layers at each level of the hierarchy, have shown limited success. The understanding developed here may help explain why: such approaches often assume that fine-grained information is uniformly helpful, without accounting for the geometric alignment between tasks. More sophisticated methods that adapt to dataset-specific alignment properties might prove more effective.

More broadly, this work connects to a fundamental question in machine learning: is it better to train a specialized model with many examples labeled at the target resolution, or to first learn from a richer supervision signal and then adapt? Training

on fine-grained labels can be viewed as a form of pre-training, but within the same data rather than transferring from a different dataset. The conditions under which this internal pre-training helps likely parallel those governing the success of transfer learning more generally.

Chapter 3

How Do Networks Respond to Natural Image Statistics?

3.1 Introduction

The retina is the first stage of visual processing in the vertebrate brain. Light entering the eye is transduced by photoreceptors and then processed through several layers of interneurons before reaching the retinal ganglion cells, whose axons form the optic nerve and carry visual information to higher brain areas. Despite its peripheral location, the retina performs sophisticated computations: different types of ganglion cells extract distinct features from the visual scene, including edges, motion, and changes in illumination. This retinal architecture shares key principles with convolutional neural networks. Both systems process information through local receptive fields, applying similar operations across spatial locations. Both use cascades of linear filtering followed by nonlinear transformations. And both build increasingly abstract representations through successive processing stages. These parallels have motivated comparisons between CNNs and the visual system more broadly, with researchers finding that networks trained on object recognition develop representations that predict neural responses in visual cortex [27, 28]. Yamins and colleagues showed that hierarchical convolutional networks optimized for categorization tasks, without any explicit constraints on neural activity, nonetheless develop response properties that match those recorded from neurons in V4 and inferotemporal cortex [27]. Subsequent work extended these findings, demonstrating that the layer structure of deep networks corresponds to the hierarchical organization of visual areas: early layers match early visual cortex, while deeper layers match higher areas along the ventral stream [29, 30]. This correspondence has made CNNs useful tools for understanding visual processing, providing computational models that can be probed and manipulated in ways that biological systems cannot.

Most of this comparative work has focused on cortical areas, particularly V1, V4,

and inferotemporal cortex. The retina has received less attention, despite being the first stage of visual processing and performing its own sophisticated computations that shape all downstream representations. One reason may be that the retina is often viewed as a relatively simple preprocessing stage, but this underestimates the complexity of retinal circuits. Retinal ganglion cells come in many types, each extracting different features, and their collective output represents a highly processed version of the visual input rather than a raw image. A key insight from decades of research on the retina is that its processing is adapted to the statistical structure of natural images. Natural scenes are not random: nearby pixels tend to have similar intensities, and this spatial correlation extends over substantial distances. Quantitatively, natural images exhibit a characteristic power spectrum where amplitude falls approximately as $\frac{1}{f^\alpha}$, with α close to 2 [31, 32]. This power law reflects the scale-invariant structure of natural scenes, where similar patterns of edges and textures appear at multiple spatial scales. The exponent α characterizes the correlation structure: higher values correspond to stronger long-range correlations and smoother images, while lower values correspond to more high-frequency content.

Prior work has established that the power spectrum plays an important role in early visual processing. Natural images, with their characteristic $1/f^\alpha$ spectrum and $\alpha \approx 2$, contain spatial redundancies that the retina appears adapted to handle [33, 34, 31]. But most of this work has focused on how the visual system transforms its inputs. A simpler question remains: does the power spectrum also influence how strongly visual systems respond to different images?

To address this, we examined whether the power spectrum affects retinal activation using a CNN trained to reproduce retinal ganglion cell responses [35, **?**]. Given the strong architectural parallels between retinal circuits and CNNs, we also tested VGG16 [36], a standard convolutional network trained on ImageNet. This network serves as the artificial equivalent of a system exposed to natural images, much like the biological retina. We generated synthetic images with power spectra $1/f^\alpha$, varying α from 0 to 5, with randomized phase to isolate the effect of the amplitude spectrum from any structured spatial content.

We found that both models showed similar response patterns: mean activation decreased with α following a sigmoidal curve, with the steepest decline, and therefore peak sensitivity, occurring near $\alpha \approx 2$, precisely the natural image regime. But why? One possibility is that training on natural images shapes networks to be preferentially sensitive to these statistics. Another possibility is that this sensitivity arises from the convolutional architecture itself, independent of learning.

To distinguish these hypotheses, we tested randomly initialized convolutional networks that had never been trained. A simple random CNN reproduced the same sigmoidal response pattern, while a random fully connected network showed flat re-

sponses across all α values. This demonstrates that spectral sensitivity requires the spatial structure of convolution, not learning. We then asked what determines the critical α where sensitivity peaks, finding that it scales with input dimensionality: 1D convolutions peak near $\alpha \approx 1$, 2D near $\alpha \approx 2$, and 3D near $\alpha \approx 3$. Finally, we investigated whether this architectural property has functional consequences by training networks on modified versions of CIFAR-10 with different spectral slopes. We found that learning efficiency depends on the spectral regime: classification accuracy was highest in the range where the network shows greatest sensitivity, while performance declined at both very low and very high α values where network responses saturate. Notably, the best accuracy exceeded that of the original dataset, and this optimal spectral slope coincided approximately with the network's critical α. This suggests that architectural properties may shape learning efficiency, though the full implications of this relationship remain to be explored.

3.2 Related Works

3.2.1 Statistics of Natural Images

Understanding how images from the natural world are statistically structured has been a long-standing question in vision science. Early work by Field [31] examined whether the properties of cortical neurons might reflect an adaptation to the statistical regularities present in natural scenes. In his seminal 1987 paper, Field analyzed an ensemble of six images taken in wooded environments and discovered that their amplitude spectra consistently fall off as $1/f$, meaning the power spectra decrease as $1/f^2$. This relationship implies that natural images have equal energy per octave, a form of scale invariance where the statistical structure looks similar whether you examine fine details or coarse features.

Field's key insight was connecting these statistics to neural coding. He proposed that cortical simple cells, which have spatial frequency bandwidths around 1 octave, are particularly well matched to natural image statistics. When the image's power spectrum falls off as $1/f^2$, using channels with constant octave bandwidths means that each channel receives roughly equal energy on average. This converts the higher-order redundancy present in natural images, where neighboring pixels are correlated in complex ways, into first-order redundancy, where most sensors are inactive but a sparse subset responds strongly. Field argued this transformation is precisely what you want for efficient coding, since it allows the visual system to represent scenes using only a small fraction of active neurons at any moment.

Ruderman and Bialek [32] extended this line of investigation in 1994 with a more systematic study using 45 images from woodland scenes. They examined not just

global spectral properties but also the distribution of local quantities like contrast measured at different spatial scales. Their work confirmed that natural images exhibit scaling behavior, though with a slightly different characterization. They found the power spectrum follows $S_\phi(k) = A/k^{2-\eta}$ with $\eta \approx 0.19$, which is close to but not exactly $1/f^2$. More importantly, they discovered that the probability distributions of local contrast have nearly exponential tails rather than Gaussian ones, and these distributions remain similar across scales, another signature of scale invariance.

One of Ruderman and Bialek's most intriguing findings was that variance normalization, where you divide local measurements by the local standard deviation, transforms the non-Gaussian distributions of natural images into approximately Gaussian ones. This operation mimics the gain control mechanisms observed in biological vision systems and suggests that the visual system actively normalizes for local contrast variations. After this normalization, images become more Gaussian, meaning they have maximal entropy given their variance. This implies that the non-Gaussian structure of raw natural images encodes important information, and the visual system needs to actively process this structure rather than simply measuring local luminance values.

Subsequent large-scale analyses revealed that these spectral regularities are even more nuanced than initially thought. Torralba and Oliva [37] analyzed approximately 12,000 images spanning diverse scene categories and found that while the overall $1/f^\alpha$ relationship holds with $\alpha \sim 2$ (or $\alpha \sim 1$ for amplitude spectra), the precise value of the exponent varies systematically with both orientation and scene type. They proposed a more complete model where the expected power spectrum depends on orientation: $E[|I(f,\theta)|^2] \approx A_s(\theta)/f^{\alpha_s(\theta)}$, with both the amplitude factor $A_s(\theta)$ and the frequency exponent $\alpha_s(\theta)$ varying as functions of orientation.

This orientation dependence reflects a striking anisotropy in natural images. Vertical and horizontal orientations are substantially more frequent than obliques across both natural landscapes and man-made environments, though the pattern differs between these categories. For natural scenes, the exponent α averages around 1.98 for horizontal orientations, 2.02 for obliques, and 2.22 for vertical orientations. Man-made scenes show a different pattern, with α values of 1.83 for horizontal, 2.37 for obliques, and 2.07 for vertical orientations. The amplitude factors similarly vary, creating characteristic spectral signatures that distinguish scene categories. Open landscapes like beaches and coasts show strong horizontal structure from the horizon line, while indoor scenes and urban environments exhibit more balanced energy across orientations, often with a cross-shaped pattern from the prevalence of vertical and horizontal edges in architectural structures.

These spectral signatures also vary systematically with scene scale, the mean distance between the observer and the principal scene elements. For man-made environments viewed at close range (1-5 meters), power is concentrated at low spatial

frequencies corresponding to smooth surfaces. As viewing distance increases and the field of view encompasses more objects and details, energy at higher frequencies grows, reflecting increased clutter and texture. Natural environments show the opposite trend. Close-up views of natural textures like bark or leaves have substantial high-frequency content, but as distance increases, fine details blur and large-scale structure like mountain profiles or forest canopies dominates, shifting energy toward lower frequencies. This difference reflects how man-made and natural structures are organized at different spatial scales.

Both papers emphasize that natural images are far from random. They have consistent statistical structure that can be exploited for efficient representation. The $1/f$-like power spectrum, the heavy-tailed distributions of local features, the scale invariance, and the orientation anisotropy all point to strong regularities in natural scenes. These regularities likely reflect the physical processes that generate natural images, such as the way light interacts with surfaces at multiple spatial scales, from the texture of bark on a nearby tree to the outline of a distant mountain, combined with the ecological constraints on how animals like humans typically view their environment.

These findings about the spectral statistics of natural images provide the foundation for our second project (Chapter 3), where we systematically investigate how convolutional neural networks respond to stimuli with different power law exponents. By synthesizing images with controlled power spectra of the form $S(k) \propto 1/k^{\alpha}$ and varying α, we can test whether CNNs show preferential sensitivity to the $\alpha \approx 2$ statistics characteristic of natural scenes, and whether this sensitivity reflects learned adaptation to natural image statistics or emerges from the architectural properties of convolutional layers themselves.

3.3 Convolutional Neural Networks as Models of the Visual System

A central motivation for using convolutional neural networks to study visual processing comes from mounting evidence that these architectures capture key organizational principles of the mammalian ventral stream. The correspondence between CNNs and biological vision is not merely superficial, modern deep CNNs trained on object recognition tasks have been shown to quantitatively predict neural responses in higher visual areas with unprecedented accuracy.

Yamins and colleagues [27] demonstrated this correspondence in a landmark 2014 study where they systematically evaluated thousands of hierarchical neural network models against recorded responses from neurons in macaque V4 and IT cortex. Their key finding was that a model's performance on challenging object recognition tasks

strongly predicted how well it could explain variance in neural responses. Using high-throughput optimization methods, they identified a CNN architecture (the HMO model) that matched human-level performance on various recognition tasks. Critically, even though this model was never explicitly fit to neural data, its output layer was highly predictive of IT responses (explaining roughly 49% of explainable variance), while intermediate layers predicted V4 responses (explaining about 52% of explainable variance). This suggests that the computational pressures that shaped biological vision, primarily the need to recognize objects under transformation, also determine the representations that emerge in performance-optimized artificial networks.

The correspondence extends beyond simple prediction accuracy to the organizational structure of representations. Güçlü and van Gerven [29] provided direct evidence for a gradient in representational complexity across the human ventral stream by showing that successive layers of a deep CNN predict responses in progressively downstream visual areas. Early layers, which learn relatively simple edge and texture features, best predicted responses in V1 and V2. Intermediate layers captured more complex conjunctions and predicted V4 responses. The deepest layers, which represent high-level object parts and whole objects, predicted responses in lateral occipital cortex and inferior temporal areas. This layer-to-area mapping was not imposed by the experimenters but emerged naturally from selecting whichever layer best predicted each voxel's responses.

The parallel between CNNs and biological vision operates at multiple levels of description. Architecturally, both systems use hierarchical processing with local computations that are repeated across space (convolution), alternating linear filtering with pointwise nonlinearities, and pooling operations that build invariance. Functionally, both systems solve the same core problem of building representations that support robust object recognition despite huge variation in viewing conditions. Representationally, the features that emerge at different depths in trained CNNs, from oriented edges to complex object-selective responses, mirror the progression observed along the ventral stream.

These findings establish CNNs as more than convenient feature extractors for neuroimaging analysis. They are computational hypotheses about how the visual system actually works. When we use a pretrained CNN to investigate how networks respond to images with different spectral statistics, we're not just studying an arbitrary algorithm. We're probing a system whose representations have been validated against biological data and whose intermediate computations reflect, at least approximately, the transformations performed by real neural circuits. This makes CNNs particularly appropriate tools for generating testable predictions about how biological vision might be tuned to the statistical structure of natural images.

3.4 Methods

3.4.1 Natural Image Datasets

To analyze whether different object categories exhibit characteristic spectral properties, we used Tiny ImageNet, a subset of the ImageNet dataset containing 200 classes with 500 training images per class. Each image was 64×64 pixels, matching the dimensions of our synthetic images. For each image, we computed its power spectrum by taking the Fourier transform and calculating $P(f) = |F(f_x, f_y)|^2$. We then fit a power law $P(f) \propto 1/f^\alpha$ to extract the spectral exponent α for each image. By comparing the distribution of α values across different object categories, we could determine whether certain classes show preferences for specific spectral slopes.

3.4.2 Synthetic Image Generation

To isolate the effect of spectral properties on neural network behavior, we generated synthetic images with controlled power spectra. Natural images typically exhibit power law statistics, where the power spectrum $P(f) \propto 1/f^\alpha$ with the spatial frequency f. By systematically varying the exponent α, we could test how different spectral characteristics affect network responses while keeping other image properties constant.

We generated images in the Fourier domain and then transformed them to spatial coordinates. For each image of size $N \times N$ pixels, we created a 2D grid of frequency coordinates (f_x, f_y) and computed the magnitude of each frequency component as:

$$f(f_x, f_y) = \sqrt{f_x^2 + f_y^2} \tag{3.1}$$

The amplitude at each frequency was set according to the desired power spectrum:

$$A(f) = \sqrt{f^{-\alpha}} = f^{-\alpha/2} \tag{3.2}$$

This relationship ensures that the power spectrum $P(f) = |A(f)|^2 \propto 1/f^\alpha$ follows the specified power law. We assigned random phases uniformly distributed between 0 and 2π to each frequency component:

$$\phi(f_x, f_y) \sim \mathcal{U}(0, 2\pi) \tag{3.3}$$

The complex Fourier coefficients were then constructed as:

$$F(f_x, f_y) = A(f) \cdot e^{i\phi(f_x, f_y)} \tag{3.4}$$

To ensure the inverse Fourier transform produces real-valued images, we enforced

Hermitian symmetry on the Fourier coefficients:

$$F(-f_x, -f_y) = F^*(f_x, f_y) \tag{3.5}$$

where $*$ denotes complex conjugation. This constraint is necessary because real-valued functions in the spatial domain correspond to Hermitian functions in the frequency domain. We set the DC component to zero, $F(0, 0) = 0$, to ensure zero mean in the spatial domain.

The spatial image was obtained through the inverse Fourier transform:

$$I(x, y) = \mathcal{F}^{-1}\{F(f_x, f_y)\} \tag{3.6}$$

where $\mathcal{F}^{-1}$ denotes the inverse Fourier transform. Finally, we normalized each image by its standard deviation:

$$I_{\text{norm}}(x, y) = \frac{I(x, y)}{\sigma_I} \tag{3.7}$$

This normalization step was important because different spectral slopes naturally produce different overall contrasts, and we wanted to isolate the effect of spectral shape rather than overall contrast level.

The resulting images shared the same spatial dimensions (64×64 pixels) and contrast but differed in their frequency content according to the specified power law exponent. This allowed us to systematically vary spectral properties while controlling for other potential confounds.

3.4.3 VGG Architecture

We used VGG16 [36], a convolutional neural network pretrained on ImageNet [3], a dataset of 1.2 million natural images spanning 1000 object categories. The architecture consists of 13 convolutional layers organized in five blocks. The first two blocks contain 2 convolutional layers each, while the last three blocks contain 3 layers each. All convolutional layers use 3×3 filters with stride 1 and padding 1, followed by ReLU activations. Each block is followed by 2×2 max pooling with stride 2. The convolutional blocks are followed by three fully connected layers with 4096, 4096, and 1000 neurons respectively.

We used the pretrained weights without fine-tuning, extracting feature representations from each of the 13 convolutional layers. For each synthetic image, we computed the mean activation across all spatial positions and channels within each layer, resulting in a single scalar value per layer that characterizes the network's response to that image.

3.4.4 Retinal Model

To study how spectral properties of images affect neural responses, we used a computational model of the retina that predicts the spiking activity of retinal ganglion cells (RGCs) in response to visual stimuli. The model, developed by Mahuas et al. [38], combines a convolutional neural network for stimulus processing with a generalized linear model (GLM) framework that captures network interactions between neurons.

Model Architecture

The model predicts the firing rate of each RGC as a function of both the visual stimulus and the recent spiking history of the local population. The stimulus processing component uses a 3D convolutional neural network with space-time separable filters. Rather than applying full 3D convolutions, the network factorizes spatial and temporal processing: each convolutional block first applies a spatial convolution across the two image dimensions, followed by a temporal convolution along the time axis. This factorization reduces the number of parameters while preserving the ability to capture spatiotemporal features in the stimulus.

The network consists of two convolutional blocks. Each block contains a space-time separable convolution, batch normalization, an exponential linear unit (ELU) activation, and adaptive average pooling. The output of the second block feeds into a factorized readout layer that produces the stimulus-driven component of the firing rate for each cell.

Network interactions are modeled through temporal filters that capture two effects: each neuron's own refractory dynamics (self-history filter) and the influence of neighboring neurons (coupling filters). These filters are represented using raised cosine basis functions, which provide a smooth, compact parameterization of the temporal dynamics. The coupling filters integrate activity over a 50 ms window, while the stimulus filters integrate over 800 ms to capture the slower dynamics of visual processing.

Two-Step Inference

A key feature of this model is the two-step training procedure that separates the inference of stimulus filters from coupling filters. When trained jointly on naturalistic stimuli with strong spatiotemporal correlations, standard GLM inference can confound stimulus-driven correlations with network-mediated correlations, leading to erroneous coupling estimates. The two-step approach solves this by first inferring coupling filters from responses to repeated stimulus presentations, where trial-to-trial variability isolates noise correlations from signal correlations. The stimulus filters are then inferred independently in a second step, and the two components are combined with appropriate corrections.

Population Simulation

A synthetic population of 49 RGCs was arranged on a triangular lattice, mimicking the mosaic organization observed in real retinae. The model for each cell uses the same CNN architecture with translated receptive fields, and neighboring cells are coupled through the inferred coupling filters based on their spatial distance. Given an input stimulus, the model generates spike trains by computing the instantaneous firing probability from the combined stimulus and network fields, then sampling spikes according to this probability.

3.4.5 Network Response Calculation

Retinal Model Responses

For each synthetic image, the model outputs the predicted spike counts for 49 ganglion cells across 5 time bins. We first computed the total spike count for each cell by summing over time bins:

$$S_i = \sum_{t=1}^{5} s_{i,t} \tag{3.8}$$

where $s_{i,t}$ is the spike count for cell i at time bin t. We then averaged across all cells to obtain a single response value per image:

$$R = \frac{1}{49} \sum_{i=1}^{49} S_i \tag{3.9}$$

This scalar value R characterizes the overall retinal response to each image, allowing us to compare sensitivity across different spectral slopes.

CNN Responses

For all convolutional neural networks (VGG16, random CNN, and dimensionality variants), we computed responses by averaging activations across the feature map after the ReLU nonlinearity. For a given layer with activation tensor $\mathbf{A}$ of shape (C, H, W) where C is the number of channels and H, W are spatial dimensions, the response to each image was:

$$R = \frac{1}{C \cdot H \cdot W} \sum_{c=1}^{C} \sum_{h=1}^{H} \sum_{w=1}^{W} A_{c,h,w} \tag{3.10}$$

For VGG16, unless otherwise specified, we report results from the final convolutional layer (block 5, layer 3) before the fully connected classification head. For the random CNN, we computed the mean activation of the single convolutional layer.

3.4.6 Power spectrum analysis and modification

Computing power spectra from natural images

To characterize the spectral properties of CIFAR-10, we computed the 2D power spectrum for each image and then averaged across the dataset. For each RGB image, we first converted to grayscale using the standard luminance weights (0.299R + 0.587G + 0.114B). We then computed the 2D discrete Fourier transform and shifted the zero-frequency component to the center. The power spectrum was calculated as the squared magnitude of this shifted Fourier transform.

To obtain a one-dimensional representation of the power spectrum, we computed the radial average by binning pixels according to their distance from the center of the Fourier domain. For an image of size $M \times N$, we calculated the distance $r = \sqrt{(x - c_x)^2 + (y - c_y)^2}$ from each pixel (x, y) to the center $(c_x, c_y) = (M/2, N/2)$, rounded to the nearest integer. We then averaged all power spectrum values at each distance r to obtain the radially averaged power spectrum $P(r)$.

Fitting power law distributions

We fit power laws of the form $P(k) \propto 1/k^\alpha$ to the radially averaged spectra by performing linear regression in log-log space. To avoid boundary effects at low frequencies and noise at high frequencies, we restricted the fit to spatial frequencies 1 through 14 (excluding the DC component at frequency 0). The slope of the linear fit in log-log coordinates directly gives the negative of the exponent α.

For CIFAR-10, we computed power spectra for all 50,000 training images and fit the radially averaged spectrum, obtaining $\alpha = 2.66$. We also computed the average power spectrum separately for each of the 10 classes and verified that all classes follow similar power law distributions with consistent exponents.

Modifying image power spectra

For each image, we applied the 2D Fourier transform separately to each color channel:

$$F(k_x, k_y) = \mathcal{F}\{I(x, y)\} \tag{3.11}$$

where $I(x, y)$ is the image intensity and k_x, k_y are spatial frequencies. We extracted the amplitude $A(k) = |F(k)|$ and phase $\phi(k) = \arg(F(k))$, then modified the amplitude spectrum:

$$A'(k) = [A(k)]^s \tag{3.12}$$

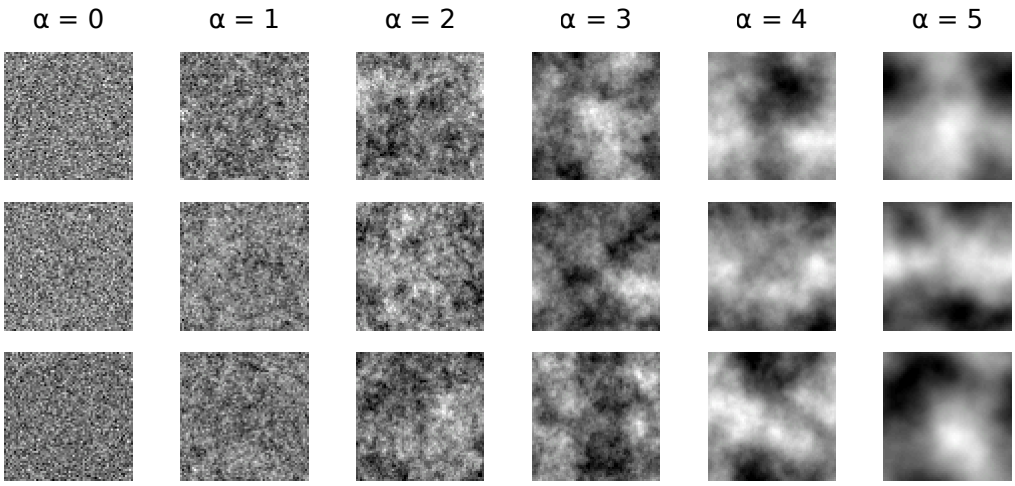

Figure 3.1: **Synthetic images with controlled power spectra.** Each column shows three example images generated with power spectra $1/f^{\alpha}$ for different values of α. At $\alpha = 0$, images resemble white noise with no spatial correlation. As α increases, images become progressively smoother and more spatially correlated. Natural images typically have $\alpha \approx 2$. Phase was randomized uniformly for all images, isolating the effect of the amplitude spectrum. Images have been standardized to zero mean and unit variance.

where s is the shift parameter. For a power spectrum originally following $P(k) = |A(k)|^2 \propto 1/k^{\alpha}$, this gives:

$$P'(k) = |A'(k)|^2 = |A(k)|^{2s} \propto 1/k^{\alpha \cdot s} = 1/k^{\alpha'} \tag{3.13}$$

with effective exponent $\alpha' = \alpha \cdot s$.

We reconstructed each color channel by combining the modified amplitude with the original phase:

$$F'(k) = A'(k) \cdot e^{i\phi(k)} \tag{3.14}$$

and applying the inverse Fourier transform. Finally, we normalized each modified image to preserve its original mean μ and standard deviation σ, ensuring that differences in network performance reflected spectral changes rather than overall brightness or contrast variations.

3.5 Results

3.5.1 Networks show peak sensitivity to natural image power spectra

To test whether power spectrum statistics affect network responses, we compared two systems: a retinal model trained to predict ganglion cell responses and VGG16 pre-

trained on ImageNet. We generated synthetic images with controlled spectral properties, with power spectra of the form $1/f^{\alpha}$. We changed α from 0 (white noise) to 5 (highly smooth, correlated images) in steps of 0.2, with phase randomized uniformly (Figure 3.1). All images were standardized to zero mean and unit variance to isolate the effect of spectral slope from overall contrast. For each value of α, we generated 200 images and measured network responses in two mentioned systems. Details of image generation and network architectures are provided in Methods.

The input images differed between models to match each system's expected format. For the retinal model, we used 64×64 pixel images, the size on which the model was trained. For VGG16, we used 224×224 images to match its ImageNet preprocessing. What we measured also differed between systems. For the retinal model, we computed the average number of predicted spikes across all 49 simulated ganglion cells (see Methods for details). For VGG16, we computed the mean activation across all nodes in the final layer before the 1000-class prediction head, as this provides a comparable summary measure to the retinal model's output.

Both systems showed a consistent pattern: mean activation decreased monotonically as α increased, following a sigmoidal curve (Figure 3.2). At low α values near white noise, responses were high. As α increased toward the natural image regime and beyond, responses dropped before saturating at high α values. The response curves for the retinal model appear noisier than those for VGG16, but this reflects the stochastic nature of the model. The retinal model incorporates a stochastic component that mimics the trial-to-trial variability observed in real retinal recordings. Given a fixed image, repeated presentations do not produce identical spike counts. We could have bypassed this stochasticity, but we wanted the model's behavior to remain faithful to real retinal physiology.

To identify where networks were most sensitive to changes in spectral slope, we computed the derivative of these response curves. The derivative captures how rapidly responses change as α varies: where it is steepest, a small shift in α produces the largest change in activation. In both systems, the derivative reached its most negative value near $\alpha \approx 2$, precisely the regime characteristic of natural images (Figure 3.2, bottom panels). We repeated this experiment in all the convolutional layers of the VGG, obtaining the same results (Appendix B.2.1). This suggests that the property observed is a general feature of the architecture, rather than something specific to particular processing stages.

3.5.2 Peak sensitivity arises from architecture, not training

The peak sensitivity near $\alpha \approx 2$ could arise from two sources: networks might learn this pattern through exposure to natural images during training, or it could be an inherent

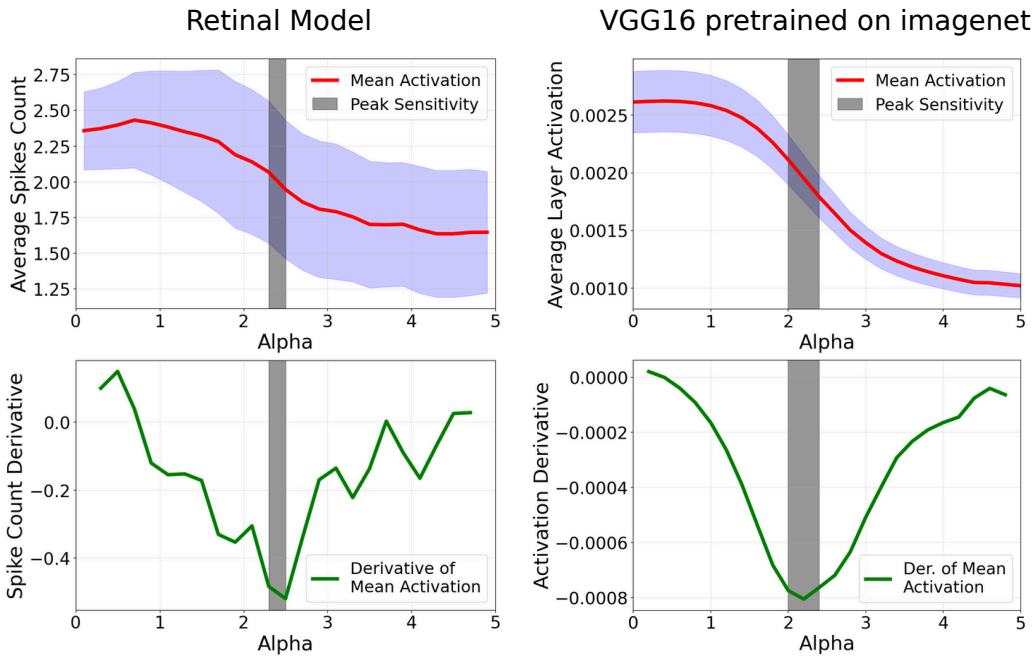

Figure 3.2: **Response and sensitivity curves respect to power spectrum exponent for two neural systems.** Top panels show mean activation as a function of α for the retinal model (left, average spike count across 49 ganglion cells) and VGG16 (right, mean activation in the final layer before classification). Red line indicates mean response across 200 images per α value, shaded region shows standard deviation. Bottom panels show the derivative of the activation curves (computed using centered finite differences), measuring how rapidly responses change with α. Gray bars mark the region of peak sensitivity (most negative derivative), which occurs near $\alpha \approx 2$ for both systems, the characteristic slope of natural images.

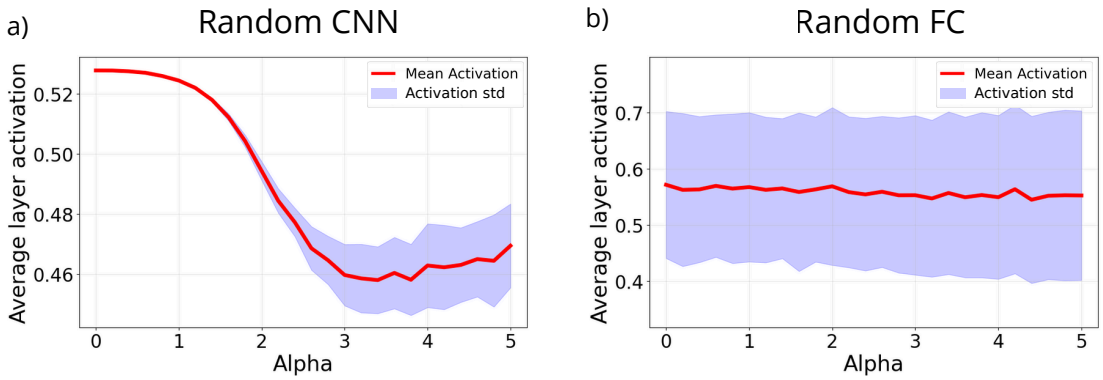

Figure 3.3: **Sigmoidal response pattern arises from convolutional architecture, not training**. (a) A randomly initialized convolutional network (single layer with 32 filters, 3×3 kernels, ReLU activation) reproduces the sigmoidal response pattern observed in trained networks. (b) A randomly initialized fully connected network (1024 neurons, ReLU activation) shows flat responses across all α values. Both networks used He normal initialization and processed the same 224×224 pixel synthetic images. Red line indicates mean activation across 200 images per α value, shaded region shows standard deviation. The contrast between panels demonstrates that the sensitivity pattern requires the spatial structure of convolution, not simply random weights and nonlinearities.

property of convolutional architectures. To distinguish between these possibilities, we tested randomly initialized networks that had never been trained.

We compared two types of random networks processing the same 224×224 pixel synthetic images with varying α. The first was a simple convolutional network consisting of a single convolutional layer (32 filters, 3×3 kernels, matching the first layer of VGG16) followed by ReLU activation. The second was a fully connected network with a single hidden layer of 1024 neurons. Both networks used He normal initialization [39]. For the random CNN, we measured mean activation across all units in the convolutional layer. For the fully connected network, we measured mean activation across all units in the hidden layer.

The random CNN reproduced the sigmoidal response pattern observed in trained networks (Figure 3.3a). Mean activation decreased monotonically as α increased, dropping sharply in the natural image regime before saturating at high α values. In contrast, the random fully connected network showed nearly flat responses across all α values (Figure 3.3b). The flat response of the fully connected network shows that the sigmoidal pattern depends on the convolutional structure itself, not on learned weights. Convolution operates through local linear combinations, and the pattern likely emerges from how these operations interact with the spatial correlation structure of images at different α values. Fully connected networks, which lack this spatial locality, do not exhibit this behavior.

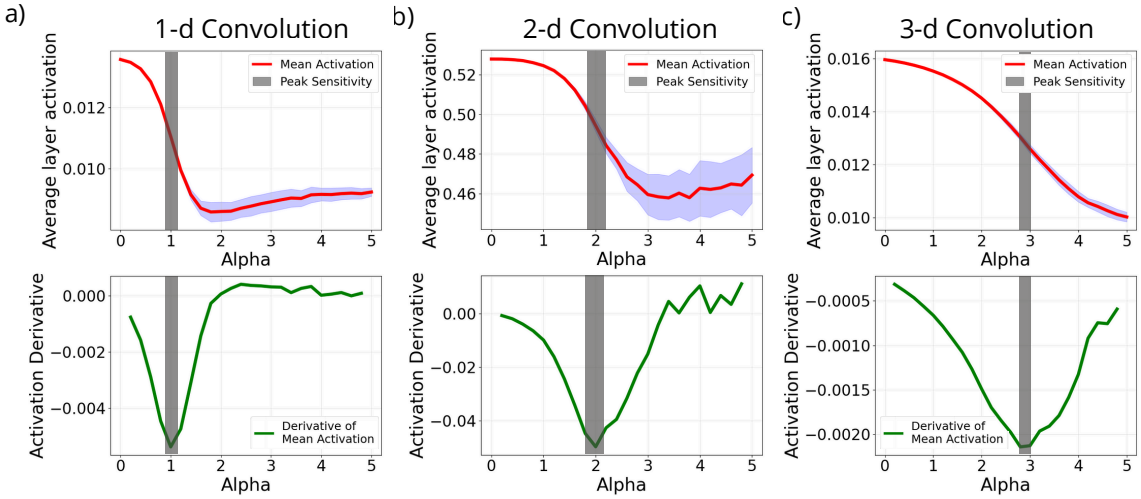

Figure 3.4: **Response of random convolutional filters to synthetic signals across different input dimensionalities.** Top row: mean activation of a single convolutional layer with random weights, plotted as a function of the spectral exponent α for one-dimensional signals (a), two-dimensional images (b), and three-dimensional volumes (c). Bottom row: derivative of the activation curves, identifying where sensitivity to changes in α is greatest. The gray shaded region marks the peak sensitivity in each case. For all three dimensionalities, the response follows a sigmoidal curve similar to that observed in trained networks, but the location of peak sensitivity shifts systematically with input dimension: the 1D convolution peaks near $\alpha \approx 1$, the 2D convolution near $\alpha \approx 2$, and the 3D convolution near $\alpha \approx 3$. Kernels were initialized using Kaiming uniform initialization. For each α value, we averaged responses over 100 synthetic signals and 32 random kernel initializations. Shaded regions indicate standard error.

3.5.3 Peak sensitivity scales with input dimensionality

Having established that peak sensitivity arises from convolutional architecture rather than training, we asked what determines where this peak occurs. For images, all networks we tested showed peak sensitivity near $\alpha \approx 2$, the characteristic exponent of natural images. But is this value specific to 2D convolutions, or would it hold for convolutions operating on inputs of other dimensionalities?

To test this, we generalized our synthetic stimulus generation to one and three dimensions. For 1D signals, we generated vectors of length 16000 (comparable to typical audio samples) with power spectra $P(f) \propto 1/f^{\alpha}$, using the same random-phase procedure as for images. For 3D volumes, we generated $32 \times 32 \times 32$ cubes following the analogous construction in three-dimensional Fourier space. We then measured responses from single-layer convolutional networks with random weights, matching the architecture of our 2D experiments but with 1D or 3D convolutions as appropriate. Kernels were initialized using He Normal initialization, and we averaged responses over 200 signals per α value and 32 random kernel initializations.

The results revealed a clear pattern (Figure 3.4). All three networks showed the same sigmoidal response curve we observed earlier, but the location of peak sensitivity shifted systematically with dimensionality. The 1D convolution showed peak sensitivity near $\alpha \approx 1$, the 2D convolution near $\alpha \approx 2$, and the 3D convolution near $\alpha \approx 3$. In each case, the critical α matched the dimensionality of the input.

This scaling can be understood by considering what happens when a power-law signal is convolved with an ideal filter in d dimensions. In the Fourier domain, the output power of a convolution is the input power spectrum weighted by the filter's squared frequency response. In one dimension, for an input with $P(k) \propto k^{-\alpha}$ and an ideal lowpass filter $|H(k)|^2 = \mathbf{1}_{[0,F]}(k)$, the total output power is

$$P_{\mathrm{LP}_1}(\alpha) = \int_0^F k^{-\alpha}\,\mathrm{d}k \tag{3.15}$$

In d dimensions, the corresponding ideal lowpass filter passes all frequencies inside a ball of radius F: $|H(\mathbf{k})|^2 = \mathbf{1}_{|\mathbf{k}|\le F}$. The output power is an integral over d-dimensional Fourier space, which we evaluate by switching to spherical coordinates. The volume element becomes $k^{d-1}\,dk\,d\Omega$, and since both the power spectrum $k^{-\alpha}$ and the filter $\mathbf{1}_{[0,F]}(k)$ depend only on $|\mathbf{k}|$, the angular integral contributes a constant Ω_{d-1} (the surface area of the unit sphere in d dimensions). The result is

$$P_{\mathrm{LP}_d}(\alpha) = \Omega_{d-1}\int_0^F k^{-\alpha}\,k^{d-1}\,\mathrm{d}k = \Omega_{d-1}\int_0^F k^{-(\alpha-d+1)}\,\mathrm{d}k = \Omega_{d-1}\,P_{\mathrm{LP}_1}(\alpha-d+1) \tag{3.16}$$

The only difference between the 1D and the d-dimensional case is the k^{d-1} factor from the Jacobian of the volume element, which shifts the exponent by $d-1$. The prefactor Ω_{d-1} rescales the curve vertically without changing its shape or the location of its peak. The same argument applies to any ideal filter defined by a condition on $|\mathbf{k}|$ alone, including highpass and bandpass filters. In all cases, the d-dimensional output power curve is a shifted copy of the one-dimensional curve: $P_d(\alpha) \propto P_1(\alpha-d+1)$.

This shift in output power translates directly to a shift in the activation curve after the ReLU nonlinearity. The preactivation of each convolutional unit is zero-mean, since the input signal has zero DC component and convolution is linear. Its variance is therefore the output power $P_d(\alpha)$. As we verify numerically in Appendix B.1.1, the mean activation after ReLU is proportional to the standard deviation of the preactivation, so

$$\mathrm{activation}_d(\alpha) \;\propto\; \sqrt{P_d(\alpha)} \;\propto\; \sqrt{P_1(\alpha-d+1)} \;\propto\; \mathrm{activation}_1(\alpha-d+1) \tag{3.17}$$

The entire activation curve, including where its derivative peaks, shifts by $d-1$ in α. We demonstrate this for ideal spherically symmetric filters, for which the identity

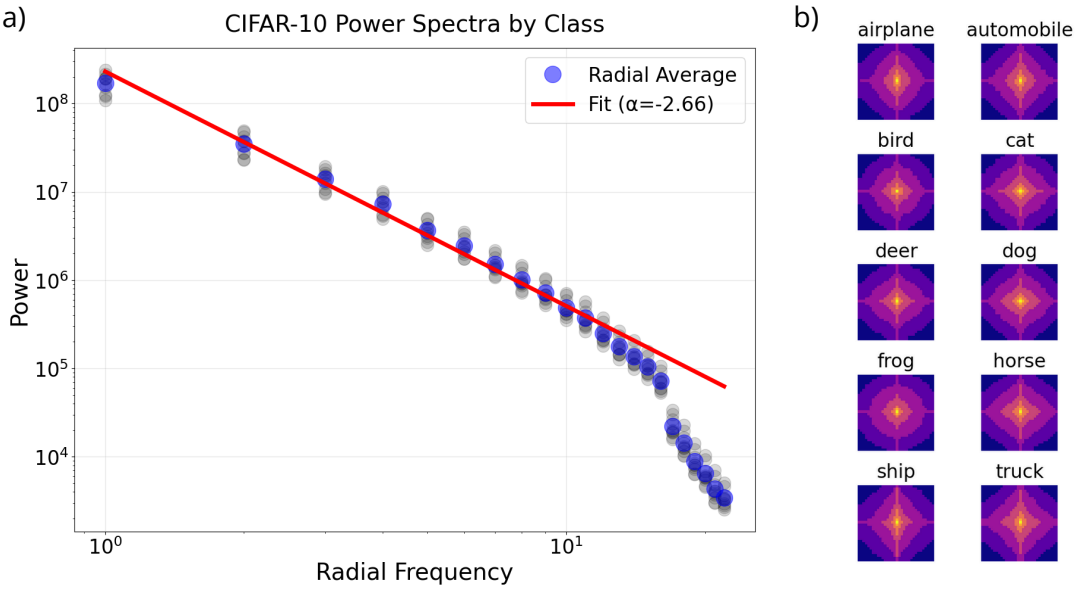

Figure 3.5: **CIFAR-10 exhibits characteristic natural image power spectra.** (a) Radially averaged power spectra as a function of spatial frequency. Gray points show individual classes, blue points show the average across the entire dataset, and the red line shows a power law fit ($P(k) \propto 1/k^{\alpha}$) with $\alpha = 2.66$. The fit was performed over the first 14 frequencies to exclude and high-frequency noise. Individual class spectra collapse onto the overall dataset spectrum. (b) Two-dimensional power spectra averaged across all images within each CIFAR-10 class. Power is concentrated at low frequencies (image center) and declines toward higher frequencies. The color scale is discretized into 7 levels, which emphasizes the appearance of class differences, though the underlying spectra are highly similar across categories.

is exact. The same shift is observed experimentally for random convolutional filters (Figure 3.4), consistent with the calculation. Since we observe the peak at $\alpha^* \approx 1$ in 1D, the shift predicts $\alpha_{\text{critical}} \approx d$ in d dimensions, which is exactly what we find. We return to the question of why the one-dimensional baseline sits at $\alpha^* \approx 1$ in the Discussion.

3.5.4 Does spectral sensitivity matter for natural image classification?

We established that untrained convolutional networks show peak sensitivity to specific spectral slopes, with this critical α value matching the network's dimensionality. However, observing sensitivity in average activations doesn't tell us whether this architectural preference matters for learning. Does training on images that match the network's preferred α actually improve performance?

To answer this question, we turned to CIFAR-10, a standard benchmark dataset of natural images. Like other natural image datasets, CIFAR-10 exhibits power spectra

that follow a power law distribution [31, 32, 37]. We computed the average power spectrum across the entire dataset and fit a power law to the radially averaged spectrum (Figure 3.5a). The fit yielded $\alpha \approx 2.66$, consistent with the typical $1/f^{\alpha}$ statistics of natural images where α falls between 2 and 3.

While all classes in CIFAR-10 show similar power law behavior, we can observe slight class-specific variations in their spectral properties (Figure 3.5b). The overall structure remains consistent across categories, with power concentrated at low frequencies (the bright center) and declining toward higher frequencies. When radially averaged, these spectra collapse onto a common power law (Figure 3.5a, gray points show individual classes), consistent with the overall dataset fit of $\alpha \approx 2.66$.

Having established that CIFAR-10 has a characteristic spectral slope of $\alpha \approx 2.66$, we designed an experiment to test whether the dataset's spectral properties influence classification performance. If the architectural sensitivity we observed in untrained networks has some influences on the learning process, then training on images with different spectral slopes might affect how well networks learn to classify them. Specifically, we hypothesized that learning might be more effective around the sensitivity peak ($\alpha \approx 2$ for 2D networks), where the network's activations are most responsive to changes in spectral properties. In contrast, at very low α or very high α, where sensitivity is low and activations approach saturation, the network might struggle to distinguish between inputs, potentially complicating learning.

To test this, we created modified versions of CIFAR-10 spanning a range of spectral slopes. Our approach preserved each image's phase information while adjusting its amplitude spectrum. For each image, we raised the Fourier amplitude at each frequency to a power s. This transformation shifts the spectral slope: if the original power spectrum follows $P(k) \propto 1/k^{\alpha}$, the modified spectrum follows $P(k) \propto 1/k^{\alpha'}$ with $\alpha' = \alpha \cdot s$ (see Methods for details).

This procedure has an advantage over directly imposing artificial power spectra: it preserves the natural deviations from perfect power-law behavior that real images exhibit, including variations across the image and non-uniformities in the frequency distribution. Using different values of s, we created multiple versions of CIFAR-10 with spectral slopes from approximately $\alpha = 0$ to $\alpha = 4$. We trained networks from scratch on each version using identical architectures and training procedures.

The results showed a clear relationship between spectral properties and classification performance. We used a convolutional network with three convolutional blocks (32, 64, and 128 filters with 3×3 kernels, followed by batch normalization, ReLU activation, and 2×2 max pooling), a dropout layer, a dense hidden layer, and a final output layer for the 10 classes (see Figure 3.6 caption for details).

Figure 3.6a shows how images appear at different spectral slopes. At $\alpha = 0$, images are dominated by high frequencies and appear noisy, even if the original phases still

allow to see some broad shape. As α increases, images progressively lose high-frequency detail and become smoother. The original CIFAR-10 images ($\alpha = 2.66$) and images at $\alpha = 2.0$ appear similar, though the latter retain slightly more fine detail.

We trained the network on each modified dataset and measured test accuracy as a function of spectral slope (panel b). Classification accuracy followed an inverted-U shape, peaking at $\alpha = 2.1$ with 73.9% accuracy. Performance declined for both lower and higher spectral slopes, dropping to 63.0% at $\alpha = 0$ and 61.0% at $\alpha = 4.0$. Notably, the original CIFAR-10 ($\alpha = 2.66$) achieved 72.6% accuracy, lower than the peak performance. The range of high accuracy spans approximately $\alpha = 1.7$ to $\alpha = 2.5$, corresponding to what we might call the "natural" spectral regime for 2D images.

To characterize the network's intrinsic spectral preferences, we analyzed the same architecture at random initialization. We fed synthetically generated images with varying α through the untrained network and computed average layer activation and its derivative with respect to α (panel c). Both measures confirmed that the network exhibits peak sensitivity at $\alpha \approx 2.0$. Interestingly, this value falls close to the spectral slope that produced optimal classification performance, though whether this relationship holds more generally requires further investigation.

These results demonstrate that spectral properties substantially affect learning efficiency. Performance is highest in the natural spectral regime ($\alpha \approx 1.7 - 2.5$), with accuracy declining in both saturated regimes (very low or very high α). The original CIFAR-10, despite being a natural image dataset, does not yield the best classification performance, with a modest improvement possible by adjusting the spectral slope toward lower α values.

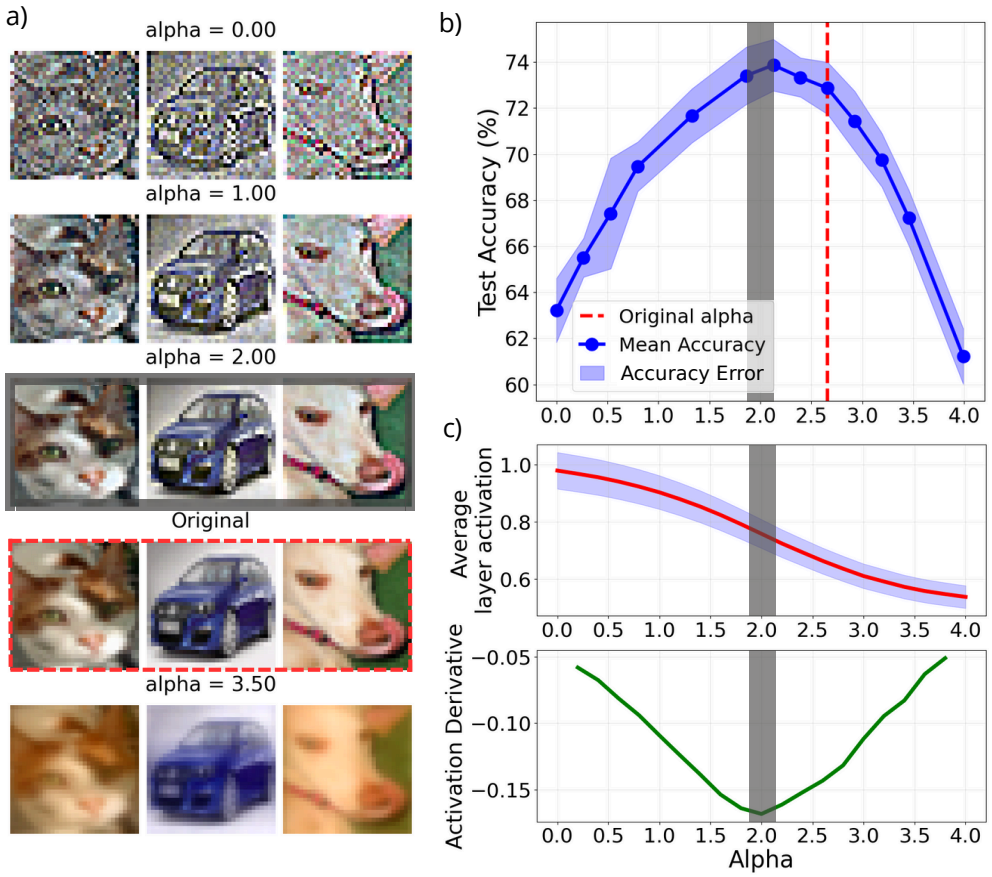

Figure 3.6: **Dataset spectral properties influence classification performance.** (a) Example CIFAR-10 images modified to have different spectral slopes. Images with $\alpha = 0$ appear noisy with dominant high frequencies, while higher α values produce progressively smoother images. The red box highlights the original dataset ($\alpha = 2.66$), corresponding to the red dashed line in panel b. The gray box highlights $\alpha = 2.0$, corresponding to the network's sensitivity peak shown in panel c. (b) Test accuracy as a function of training data spectral slope. Networks were trained on modified CIFAR-10 versions spanning $\alpha = 0$ to $\alpha = 4.0$. Accuracy peaks at $\alpha = 2.1$ (73.9%), outperforming the original dataset ($\alpha = 2.66$, 72.6%, red dashed line). Shaded region shows standard deviation across 30 independent training runs. Gray band highlights the range around the network's peak sensitivity ($\alpha \approx 2.0$). (c) Untrained network sensitivity to spectral slope. Upper panel shows average layer activation as a function of α for synthetic images, with peak at $\alpha \approx 2.0$. Lower panel shows the derivative of activation with respect to α, confirming the critical value. The gray band marks the same α range as in panel b. Training details: Adam optimizer (learning rate 0.001), batch size 64, 20 epochs. For each α value, we selected the model with highest validation accuracy across training.

3.6 Discussion

This work began with a question motivated by several observations in the literature. Natural images exhibit characteristic power law statistics, with power spectra that fall approximately as $1/f^{\alpha}$ where $\alpha \approx 2$ [31, 32]. The retina appears to be adapted to these statistics: decades of work on efficient coding has shown that retinal ganglion cells effectively decorrelate the spatial redundancies present in natural scenes [33, 34]. At the same time, there are strong parallels between retinal processing and convolutional neural networks. Both systems share local receptive fields, weight sharing across spatial locations, and cascades of linear filtering followed by nonlinearities. Both are trained, in different senses, on natural images: the retina through evolution and development, CNNs through supervised learning on datasets like ImageNet.

These parallels suggested a question: is there a deeper connection between power law statistics and the behavior of these systems? Specifically, does the power spectrum of an image directly influence how strongly networks respond to it? To investigate this, we isolated the power spectrum from other image properties by generating synthetic images with controlled spectral slopes and randomized phase. We then measured how network responses varied as a function of the spectral exponent α.

Our initial expectation was that trained networks might show a peak in activation around $\alpha \approx 2$, the regime corresponding to natural images. Since both the retinal model and VGG16 were trained on stimuli with these statistics, it seemed plausible that they would respond most strongly to inputs matching their training distribution. Instead, we found something different. Both systems showed a sigmoidal response curve: high activation at low α values, declining monotonically as α increased and images became smoother. Rather than a peak in activation at natural α values, we observed a peak in the derivative of the activation curve, meaning that sensitivity to changes in α was maximal near $\alpha \approx 2$. This pattern held for both the retinal model, despite its stochastic output reflecting the trial-to-trial variability of real retinal recordings, and for VGG16 across all convolutional layers we examined.

The role of architecture versus learning

The consistent sensitivity peak near $\alpha \approx 2$ could arise from two sources. Networks might learn this pattern through exposure to natural images during training, or it could be a structural property of the convolutional architecture itself. To distinguish between these possibilities, we examined randomly initialized networks that had never seen any training data.

The results were clear: a single convolutional layer with random weights reproduced the same sigmoidal response pattern and the same peak sensitivity near $\alpha \approx 2$. In contrast, a fully connected network with random weights showed flat responses

across all spectral slopes. This dissociation confirms that the sensitivity pattern arises from the spatial structure of convolution, not from learned weights or the presence of nonlinearities alone. Convolution operates through local linear combinations, and the sigmoidal response pattern emerges from how these operations interact with the spatial correlation structure of images at different α values.

Why does the peak occur at $\alpha = d$?

In the Results we showed that for ideal spherically symmetric filters, the activation curve of a d-dimensional convolution satisfies $\text{activation}_d(\alpha) \propto \text{activation}_1(\alpha - d + 1)$: the d-dimensional curve is a shifted copy of the one-dimensional curve, with the shift set by the k^{d-1} Jacobian from integrating over spherical shells in Fourier space. This explains why the peak sensitivity moves from $\alpha \approx 1$ in one dimension to $\alpha \approx d$ in d dimensions. What it does not explain is why the one-dimensional peak sits at $\alpha^* \approx 1$ in the first place. A full answer would require deriving the shape of the activation curve for a 1D convolution with a random filter followed by a ReLU, which depends on the distribution of the filter's frequency response and on how the nonlinearity interacts with the filtered signal. We have not carried out this derivation, and it remains an open problem.

The filter calculation leaves open why the one-dimensional baseline is at $\alpha^* \approx 1$, but the mathematical structure of the derivation offers a clue. The same k^{d-1} factor that enters as the Jacobian in the filter calculation also acts as a mode-counting factor for the signal's energy. In d dimensions, the Fourier modes at frequency k lie on a spherical shell whose surface scales as k^{d-1}, so the total energy density at frequency k is the power per mode times the number of modes:

$$\varepsilon(k) = k^{d-1}\, P(k) \propto k^{d-1-\alpha} \tag{3.18}$$

The behavior of $\int k^{d-1-\alpha}\, dk$ changes qualitatively at $\alpha = d$. When $\alpha < d$, the integral is dominated by its upper limit and most of the signal's energy comes from high spatial frequencies. When $\alpha > d$, the integral is dominated by its lower limit and low frequencies carry most of the energy. At $\alpha = d$, the spectral falloff exactly compensates the growing mode count, the integrand becomes k^{-1}, and the energy accumulates at the same rate across all multiplicative intervals of frequency: each octave contributes equally, regardless of its center frequency. This is Field's equal-energy-per-octave condition [31], which he showed holds approximately for natural images in two dimensions. Our filter calculation generalizes this: the equal-energy condition is $\alpha = d$ in any number of dimensions.

Whether this crossover in the energy distribution is the reason the activation curve transitions most steeply at $\alpha = d$ is something we cannot establish from our results, and

a complete derivation connecting the energy crossover to the shape of the activation curve remains an open problem. A derivation of the energy distribution result is given in Appendix B.1.2.

Implications for learning

Our initial hypothesis, that networks would show peak activation at the spectral slope of their training data, turned out to be wrong. The sensitivity peak at $\alpha \approx 2$ is architectural rather than learned. However, this does not mean that the architectural sensitivity has no functional consequences. Even though the network does not learn to be maximally sensitive at $\alpha \approx 2$, the fact that its architecture is tuned to this regime could still influence how effectively it learns from images with different spectral properties.

The reasoning is as follows. In the regime where sensitivity is high, small changes in α produce large changes in activation. This means the network could more readily distinguish between images that differ in their spectral properties. Conversely, in the saturation regimes at very low or very high α, activations change little with spectral slope, potentially making it harder for the network to discriminate between different inputs. If this matters for learning, we would expect classification performance to be best for datasets with spectral slopes near the sensitivity peak, and worse for datasets at saturation values of α.

We tested this prediction using CIFAR-10. The original dataset has a characteristic spectral slope of approximately $\alpha = 2.66$, consistent with other natural image datasets. We created modified versions of CIFAR-10 spanning a range of spectral slopes by adjusting the amplitude spectrum of each image while preserving its phase structure. We then trained identical networks on each version and measured classification accuracy.

The results supported our prediction. Classification accuracy followed an inverted-U shape as a function of spectral slope, peaking at $\alpha \approx 2.1$ with 73.9% accuracy. This value is close to the network's intrinsic sensitivity peak of $\alpha \approx 2.0$, as measured in the untrained network using synthetic images. Performance declined at both lower and higher spectral slopes, dropping to around 62% at the extremes. Interestingly, the original CIFAR-10 dataset, with $\alpha = 2.66$, achieved 72.6% accuracy, which is lower than the peak performance at $\alpha = 2.1$.

This last observation deserves emphasis. The original CIFAR-10 images, despite being natural images, do not produce the best classification performance for this particular architecture. A modest preprocessing step that shifts the spectral slope toward the network's sensitivity peak yields a 1.3 percentage point improvement in accuracy. Given that we averaged over 30 independent training runs, this difference is statistically meaningful.

The finding that shifting a dataset's spectral properties can improve classification

performance opens up potential applications in data preprocessing. If networks have an optimal spectral regime determined by their architecture, then adjusting training data to match this regime could be a simple way to improve learning efficiency. The effect we observed was modest but consistent, and it emerged from a single architectural choice. Different architectures, with different filter sizes, depths, or pooling strategies, might have different optimal spectral regimes.

Future directions

Several questions remain for future work. First, we tested only one architecture and one dataset in the classification experiments. To establish that the optimal training α consistently matches the network's sensitivity peak, we would need to repeat this analysis across multiple architectures and datasets. Different network designs might show different sensitivity peaks, and the relationship between sensitivity and learning efficiency might not hold universally.

Second, we focused on the magnitude of network activations, but other properties might also matter for learning. For instance, the geometry of internal representations, the diversity of activations across different filters, or the gradients during backpropagation could all vary with spectral slope in ways that affect training dynamics. Understanding which of these factors drives the relationship between spectral properties and classification performance would provide deeper mechanistic insight.

Finally, the connection between our findings and biological vision remains to be explored. We showed that a retinal model trained to reproduce ganglion cell responses exhibits the same sensitivity pattern as CNNs, but we did not test whether real retinal recordings show similar behavior. Recording from retinal ganglion cells while presenting synthetic images with controlled power spectra would allow a direct comparison between biological and artificial systems. If real retinae show the same $\alpha \approx 2$ sensitivity peak, it would suggest that this is a fundamental property of local spatial filtering, conserved across both evolved and engineered systems.

Chapter 4

Conclusions

This thesis set out to examine how data organization shapes neural network behavior, approaching this question through two distinct perspectives. The first concerned categorical structure: how the hierarchical arrangement of labels affects what networks learn and how well they generalize. The second concerned image statistics: how the spectral properties characteristic of natural scenes influence network responses and whether this influence stems from learning or from architecture itself.

In the introduction, I framed these as complementary views on how data and networks interact. Visual classification tasks are built on two fundamental components: the images themselves, with their complex mathematical structure, and the categorical labels that assign semantic meaning to visual patterns. Both components can be organized in different ways, and both, as it turns out, affect network behavior in ways that are not immediately obvious.

For hierarchical labels, the impact depends on the geometric alignment between fine and coarse classification boundaries, modulated by the parameter-to-data ratio. For spectral properties, the impact is shaped by convolutional architecture itself, which imposes sensitivity patterns independent of training. In both cases, the naive expectation turns out to be wrong: fine-grained training does not universally help, and convolutional networks are sensible to the spectral statistics of the input, even though this sensibility is structural and not learnt.

How could these results help us in practice? Consider the practical situation facing someone building a classification system. They have images to classify, and they need to decide how to label their training data. Should they invest in expert annotators who can provide fine-grained subclass labels, or is it better to collect more data with coarser labels? The experiments in Chapter 2 suggest that the answer depends on the interplay between three factors: how much data they have relative to their model's capacity, how difficult the target classification is, and whether the fine-grained distinctions align geometrically with the coarse task. When data is scarce and the fine classes reflect

natural groupings that share features with the coarse distinction, fine-grained labels help. Medical imaging applications, where disease subtypes often share underlying pathological features with their diagnostic categories, fit this pattern well. But when data is abundant or when the fine-grained distinctions are orthogonal to the coarse task, the additional annotation effort may be wasted or even counterproductive.

A related practical question concerns the images themselves. Natural images have characteristic power-law statistics. The experiments in Chapter 3 reveal that convolutional networks have an intrinsic sensitivity peak at spectral slopes matching their input dimensionality, around $\alpha \approx 2$ for standard 2D images. This sensitivity is architectural rather than learned: randomly initialized networks show the same pattern. The finding suggests a simple preprocessing strategy: if a dataset's spectral properties deviate substantially from the network's sensitivity peak, adjusting the power spectrum toward $\alpha \approx 2$ might improve learning efficiency. We observed an improvement of about 1.3 percentage points on CIFAR-10 when shifting the dataset toward the sensitivity peak, though the magnitude of this effect may vary across architectures and datasets. The same logic could work in reverse: rather than adjusting the data to match the architecture, one might choose or design architectures whose sensitivity profiles match the statistics of the target domain.

Limitations and future directions

Both investigations made a deliberate methodological choice: use simple datasets and architectures to understand the phenomena cleanly rather than optimize performance for specific applications. CIFAR-10, MNIST, and Fashion-MNIST are well-studied benchmarks where we know what to expect, making it easier to isolate the effects we wanted to examine. The architectures are similarly straightforward: fully connected networks with single hidden layers for the hierarchical labels work, standard VGG16 for the spectral experiments. This simplicity removes confounding factors and makes the results easier to interpret, but it also means the quantitative findings may not transfer directly to modern deep networks or complex real-world datasets.

This is less a limitation than a natural starting point. The qualitative insights, like the role of geometric alignment between fine and coarse boundaries or the architectural origin of spectral sensitivity, should hold more generally because they concern basic relationships between data properties and network structure. But testing this requires validation with deeper architectures and more complex data. For the hierarchical labels work, this means checking whether the parameter-to-data transitions and the boundary redundancy effects persist in convolutional networks. For the spectral sensitivity work, it means confirming the relationship between sensitivity peaks and learning efficiency across a broader range of datasets and architectures, and ideally testing whether real retinal ganglion cells show the same $\alpha \approx 2$ sensitivity we observed in the computational

retinal model.

Both projects are also strongly phenomenological. We identified patterns and demonstrated relationships empirically, but the mathematical foundations remain incomplete. For hierarchical labels, we characterized boundary redundancy precisely in the synthetic concentric circles dataset, where we could control the geometry explicitly, but we do not have a general definition that applies to arbitrary high-dimensional classification problems. Being able to measure or estimate boundary alignment from real datasets without exhaustive experimentation would make the findings more useful in practice. For spectral sensitivity, we showed that the critical α scales with input dimensionality and suggested a connection to the correlation function's transition at $\alpha = d$, but we did not derive this relationship analytically. Understanding why convolutional architectures are most sensitive at this transition point would put the empirical observations on stronger theoretical ground.

Appendix A

Supplementary Materials for Chapter 2

A.1 Supplementary Methods

A.1.1 t-distributed Stochastic Neighbor Embedding

t-distributed Stochastic Neighbor Embedding (t-SNE) is a dimensionality reduction technique introduced by van der Maaten and Hinton [40]. Unlike linear methods such as principal component analysis, t-SNE is designed to preserve local structure: points that are close in the high-dimensional space should remain close in the low-dimensional embedding. This makes t-SNE well suited for visualizing clusters and revealing geometric relationships in the data.

The algorithm works by first converting distances between data points into conditional probabilities that measure similarity. For high-dimensional data points x_i and x_j, the conditional probability $p_{j|i}$ represents the probability that x_i would pick x_j as its neighbor if neighbors were chosen in proportion to their probability density under a Gaussian centered at x_i:

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2/2\sigma_i^2\right)}{\sum_{k\neq i}\exp\left(-\|x_i - x_k\|^2/2\sigma_i^2\right)}. \tag{A.1}$$

The bandwidth σ_i is chosen separately for each point so that the conditional distribution has a fixed perplexity, a parameter that roughly corresponds to the effective number of neighbors. Points in dense regions get smaller σ_i values, while points in sparse regions get larger ones. To obtain a symmetric similarity measure, t-SNE defines the joint probability $p_{ij} = (p_{j|i} + p_{i|j})/2N$.

In the low-dimensional embedding space (typically two dimensions), t-SNE uses a Student t-distribution with one degree of freedom to measure similarity between the

embedded points y_i and y_j:

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|y_k - y_l\|^2)^{-1}}. \tag{A.2}$$

The heavy tails of the t-distribution allow moderately distant points in the high-dimensional space to be mapped farther apart in the embedding. This addresses the crowding problem: in low dimensions, there is not enough room to accommodate all the moderately distant points that exist in high dimensions.

The algorithm finds the low-dimensional embedding by minimizing the Kullback-Leibler divergence between the high-dimensional and low-dimensional distributions:

$$C = \mathrm{KL}(P\|Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{A.3}$$

This cost function penalizes configurations where the low-dimensional similarities q_{ij} fail to match the high-dimensional similarities p_{ij}. Because of the asymmetry of the KL divergence, t-SNE places more emphasis on preserving local structure: separating nearby points incurs a larger penalty than placing distant points close together. The embedding is optimized using gradient descent with momentum, typically for several hundred iterations.

The perplexity parameter controls the balance between local and global structure. We use the default value of 30, which works well for most datasets. t-SNE is a stochastic algorithm, so different random initializations can produce qualitatively different embeddings. The relative positions of clusters are meaningful, but the absolute positions and orientations are arbitrary. Similarly, distances between clusters in the embedding do not directly reflect distances in the original space. The algorithm excels at revealing whether clusters exist and which points belong together, but these visualizations should be interpreted qualitatively rather than quantitatively.

A.1.2 Internal Representations in Neural Networks

A neural network trained for classification does not directly map inputs to class labels. Instead, it progressively transforms the input through a sequence of layers, each producing an intermediate representation. These internal representations encode the features the network has learned are relevant for the task.

Consider a feedforward neural network with L layers. Given an input $x \in \mathbb{R}^d$, the network computes a sequence of hidden representations $h^{(1)}, h^{(2)}, \ldots, h^{(L-1)}$ before producing the final output. For a network with fully connected layers, the representation

at layer ℓ is computed as

$$h^{(\ell)} = \phi\left(W^{(\ell)}h^{(\ell-1)} + b^{(\ell)}\right), \tag{A.4}$$

where $W^{(\ell)}$ is the weight matrix, $b^{(\ell)}$ is the bias vector, $\phi(\cdot)$ is a nonlinear activation function (such as ReLU), and $h^{(0)} = x$ is the input. If the layer has n_ℓ neurons, then $h^{(\ell)} \in \mathbb{R}^{n_\ell}$.

The representation $h^{(\ell)}$ can be thought of as a new coordinate system in which the network re-expresses the input. In this space, the network has learned to emphasize features that help distinguish between classes while suppressing irrelevant variation. For a classification task with K classes, a well-trained network produces representations in which examples from the same class cluster together, while examples from different classes are separated.

In our experiments, we use convolutional neural networks that consist of several convolutional layers followed by one or more fully connected layers. The convolutional layers extract spatial features from the input images, and these features are then flattened and passed through the fully connected layers. We extract the activations of the last fully connected hidden layer, the layer immediately before the final classification layer. This representation has absorbed all the learned transformations and captures the features the network considers most relevant for discriminating between classes.

For the visualizations in Chapter 2, we extract these activations for a subset of the training data. If the final hidden layer has n neurons, each input image x^μ produces a representation vector $h^\mu \in \mathbb{R}^n$. We collect these representations to form a matrix $H \in \mathbb{R}^{N\times n}$, where N is the number of examples. This high-dimensional representation captures what the network has learned about the data's structure, but is difficult to visualize directly when n is large. We therefore use t-SNE to project these representations into two dimensions, which allows us to examine how the network organizes different classes in its internal representation space.

A.1.3 Implementation Details

For the CIFAR-10 visualizations, we trained a convolutional neural network on a four-class subset of the dataset. The network consisted of three convolutional blocks followed by two fully connected layers. Each convolutional block applied filters of increasing depth (32, 64, 64 filters respectively) with 3×3 kernels and ReLU activation. The first two blocks included 2×2 max pooling. The convolutional output was flattened and passed through a dense layer with 64 neurons and ReLU activation, which provides the representation we extract for visualization. The final output layer had 4 neurons with softmax activation.

The model was trained for 30 epochs using the Adam optimizer with a learning

rate of 10^{-3} and a batch size of 64. After training, we extracted representations for a balanced subset of 500 images per class (2000 total) by creating a truncated model that outputs the 64-dimensional activations of the penultimate layer. We then applied t-SNE using the scikit-learn implementation with 2 output components, 600 iterations, and perplexity 30. The resulting two-dimensional coordinates were plotted with points colored according to either their fine-grained or coarse-grained class labels.

A.2 Supplementary Figures

A.2.1 Fine vs coarse training across dataset sizes for representative Fashion-MNIST configurations

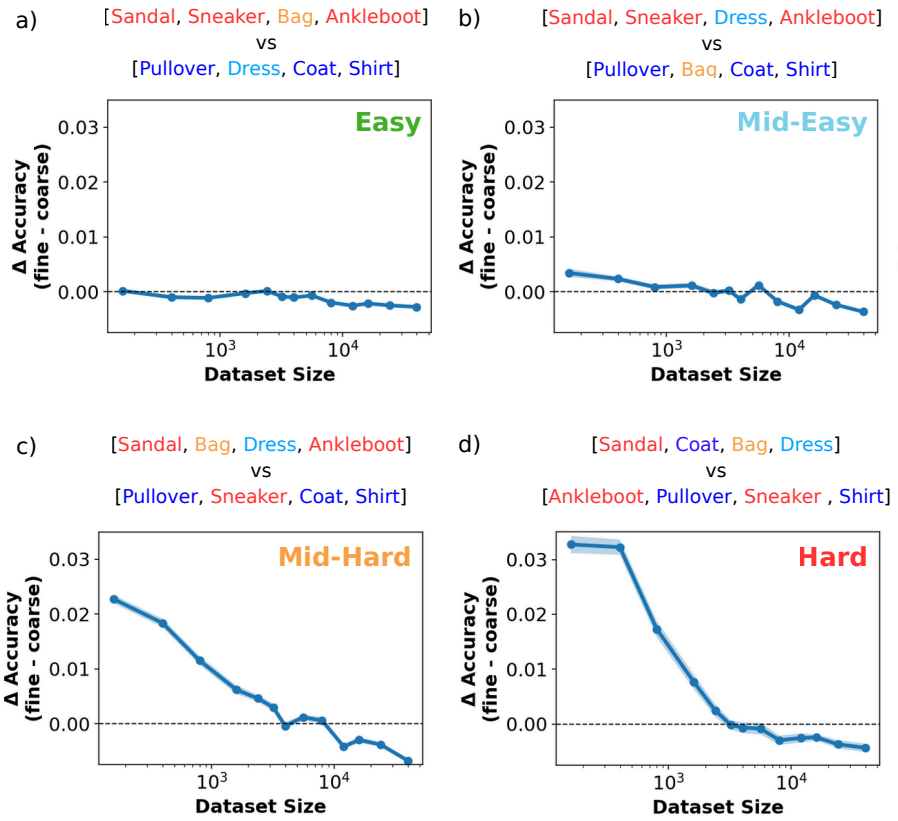

Figure A.1: **Performance gain from fine-grained training across dataset sizes for representative Fashion-MNIST configurations**. Each panel shows one configuration from the difficulty clusters identified in Figure 2.6: (a) Easy, (b) Mid-Easy, (c) Mid-Hard, and (d) Hard. The label above each plot indicates the specific class division, with colors denoting semantic groupings: reddish tones for shoe-related items and bluish tones for clothing items. Bag and Dress are shown in distinct shades within their respective blocks because they occupy intermediate positions between clusters, as indicated by silhouette score analysis and confirmed by their role in the cluster transitions. The plots demonstrate on a wider dataset size range the pattern observed in the main text: fine-grained training provides substantial and persistent benefits for hard tasks, while offering minimal or diminishing advantage for easier configurations as dataset size increases. Results are averaged over 30 independent training runs.

Appendix B

Supplementary Materials for Chapter 3

B.1 Supplementary Methods

B.1.1 Retinal Model: Technical Details

Network Architecture

The stimulus processing CNN uses depthwise separable 3D convolutions implemented as a sequence of spatial and temporal operations. The first layer applies a $1 \times 3 \times 3$ convolution (spatial only) followed by a $3 \times 1 \times 1$ convolution (temporal only). This factorization is applied in a depthwise manner, where each input channel is convolved separately before being combined.

The complete architecture parameters are:

- **First convolutional block:** 4 spatial filters per input channel, 2 temporal filters per spatial filter, 3×3 spatial kernels, temporal kernel size 3, pooling to $6 \times 12 \times 12$
- **Second convolutional block:** 2 spatial filters per input, 1 temporal filter per spatial filter, 3×3 spatial kernels, temporal kernel size 3, pooling to $3 \times 8 \times 8$
- **Readout:** Factorized as a product of spatial-temporal weights (shape $1 \times 3 \times 8 \times 8 \times N_{\text{cells}}$) and feature weights (shape $N_{\text{features}} \times N_{\text{cells}}$)

All convolutional layers use ELU activations. Batch normalization is applied after each convolutional block before the nonlinearity.

Temporal Basis Functions

The raised cosine basis functions provide a smooth parameterization of temporal filters. For a basis with n functions spanning from first peak t_1 to last peak t_2 with stretch

parameter s, the k-th basis function is:

$$\phi_k(t) = \frac{1}{2}\left(1 + \cos\left(\pi \cdot \text{clip}\left(\frac{\log(t+s) - c_k}{\Delta c}, -1, 1\right)\right)\right) \tag{B.1}$$

where c_k are the peak centers spaced linearly in log-transformed time, and Δc is the spacing between peaks.

The basis parameters used are:

- **Stimulus basis:** 8 functions, first peak at 0 ms, last peak at 720 ms, stretch 200 ms, integration window 800 ms
- **Self-history basis:** 7 functions, first peak at 0 ms, last peak at 40 ms, stretch 5 ms, integration window 50 ms
- **Coupling basis:** 7 functions, first peak at 0 ms, last peak at 25 ms, stretch 5 ms, integration window 50 ms

Spike Generation

Given stimulus input $\mathbf{S}(t)$ and spike history $\mathbf{n}(t)$, the firing probability of cell i at time t is:

$$p_i(t) = \sigma\left(h_i^{\text{stim}}(t) + h_i^{\text{self}}(t) + h_i^{\text{coupl}}(t)\right) \tag{B.2}$$

where $\sigma(x) = 1/(1+e^{-x})$ is the sigmoid function, and:

$$h_i^{\text{stim}}(t) = \text{CNN}(\mathbf{S}_{t-T:t})_i \tag{B.3}$$

$$h_i^{\text{self}}(t) = \sum_{\tau} J_{ii}(\tau) n_i(t-\tau) \tag{B.4}$$

$$h_i^{\text{coupl}}(t) = \sum_{j \in \mathcal{N}_i} \sum_{\tau} J_{ij}(\tau) n_j(t-\tau) \tag{B.5}$$

Here $\mathcal{N}_i$ denotes the neighbors of cell i determined by a distance threshold, and $J_{ij}(\tau)$ are the coupling filters reconstructed from the learned basis weights.

Synthetic Population

The synthetic population consists of 49 cells arranged on a 7×7 triangular lattice with spacing determined by the mean distance between neighboring cells in the experimental recordings (approximately 12 pixels in stimulus coordinates). Connectivity is determined by a distance threshold: cells within this threshold are coupled through the mean coupling filter estimated from the experimental data.

Simulations were run with 1 ms temporal resolution. For each stimulus presentation, 1000 repetitions were generated to estimate response statistics. Spikes were rebinned to 50 ms windows for subsequent analysis.

Mean ReLU activation and preactivation variance

The filter calculation in Section 3.5.3 establishes that, for ideal spherically symmetric filters, the output power of a d-dimensional convolution shifts as $P_d(\alpha) = P_1(\alpha - d + 1)$. To connect this to the mean activation we actually measure, we need to relate the output power (which is the variance of the preactivation, since the preactivation is zero-mean) to the mean value after the ReLU nonlinearity.

For a zero-mean random variable X with variance σ^2, the expected value of $\mathrm{ReLU}(X) = \max(0, X)$ depends on the full distribution of X. If X were Gaussian, the relationship would be $\mathbb{E}[\mathrm{ReLU}(X)] = \sigma/\sqrt{2\pi}$, giving exact proportionality between mean activation and standard deviation. The preactivation of a convolutional unit is a weighted sum of many input pixels, which by a central-limit argument should be approximately Gaussian, but this approximation is not exact in practice, particularly for small filter sizes.

To verify that the proportionality holds regardless, we tested it numerically. We generated random-phase power-law images at various values of α, convolved them with random 3×3 filters drawn from a uniform distribution on $[-1, 1]$, and measured both the standard deviation of the preactivation and the mean of the ReLU output across many filter and image realizations. The results confirm that the mean ReLU activation is proportional to the preactivation's standard deviation across the full range of α values we consider (Figure B.1). The relationship is tightly linear, validating the step from output power to mean activation used in the main text.

Given this proportionality, the chain of reasoning is as follows. The preactivation is zero-mean because the input has zero DC component and convolution is a linear operation. Its variance equals the output power $P_d(\alpha)$. Since the mean ReLU activation is proportional to the standard deviation of the preactivation, we have

$$\mathrm{activation}_d(\alpha) \;\propto\; \sqrt{P_d(\alpha)} \;=\; \sqrt{P_1(\alpha - d + 1)} \;\propto\; \mathrm{activation}_1(\alpha - d + 1) \qquad \text{(B.6)}$$

The proportionality constant does not depend on α (it depends on the shape of the preactivation distribution, which is set by the filter, not by the spectral slope). Therefore the entire activation curve, including the location of its steepest derivative, shifts by $d - 1$ when moving from one to d dimensions.

B.1.2 Energy distribution across spatial scales

Here we derive how the energy of a signal with a power-law power spectrum is distributed across spatial frequencies, and show that $\alpha = d$ is the unique exponent at which no frequency range dominates.

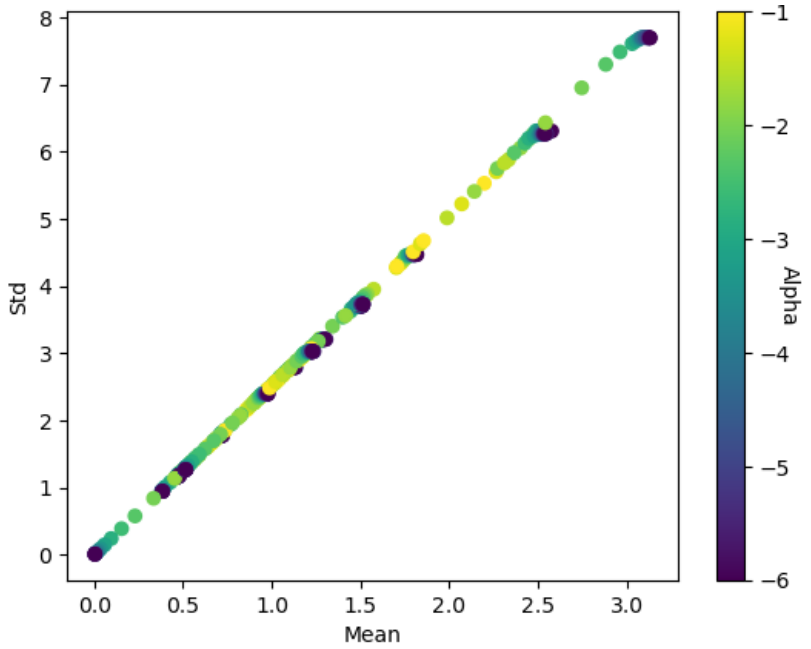

Figure B.1: **Mean ReLU activation versus preactivation standard deviation.** Each point corresponds to a different value of α, with the mean activation after ReLU plotted against the standard deviation of the preactivation before the nonlinearity, both computed across random 3×3 filters and random-phase power-law images. The tight linear relationship confirms that mean activation is proportional to the preactivation's standard deviation, validating the connection between the output power shift $P_d(\alpha) = P_1(\alpha - d + 1)$ and the shift in the activation curve.

Mode counting and the shell energy

Consider an isotropic random field in d dimensions with power spectrum $P(k) = Ak^{-\alpha}$, where $k = |\mathbf{k}|$. By Parseval's theorem, the total variance of the field is:

$$E = \int_{\mathbb{R}^d} \frac{d^d k}{(2\pi)^d} \, P(k) \tag{B.7}$$

Since $P(k)$ depends only on the magnitude k, we switch to spherical coordinates in Fourier space. The volume element is $d^d k = \Omega_d \, k^{d-1} \, dk$, where $\Omega_d = 2\pi^{d/2}/\Gamma(d/2)$ is the total solid angle in d dimensions ($\Omega_1 = 2$, $\Omega_2 = 2\pi$, $\Omega_3 = 4\pi$). The total energy becomes:

$$E = \frac{\Omega_d}{(2\pi)^d} \int_0^\infty k^{d-1} \, P(k) \, dk \tag{B.8}$$

The energy in a thin shell between k and $k + dk$ is therefore:

$$dE = \frac{\Omega_d}{(2\pi)^d} \, k^{d-1} \, P(k) \, dk \tag{B.9}$$

The factor k^{d-1} is geometric: it counts the number of independent Fourier modes in the shell. In two dimensions, the shell at radius k is a ring of circumference $2\pi k$; in

three dimensions, a sphere of surface area $4\pi k^2$. Higher frequencies have proportionally more modes. For $P(k) = Ak^{-\alpha}$, the energy density weighted by the mode count is:

$$\varepsilon(k) \equiv \frac{dE}{dk} \propto k^{d-1}\, k^{-\alpha} = k^{d-1-\alpha} \tag{B.10}$$

Which frequencies dominate the total energy

The total energy is:

$$E \propto \int_{k_{\min}}^{k_{\max}} k^{d-1-\alpha}\, dk \tag{B.11}$$

where $k_{\min} \sim 1/L$ is set by the image size and $k_{\max}$ is the Nyquist frequency. The behavior depends on $\beta = d - 1 - \alpha$:

$$\int_{k_{\min}}^{k_{\max}} k^{\beta}\, dk = \begin{cases} \dfrac{k_{\max}^{d-\alpha} - k_{\min}^{d-\alpha}}{d-\alpha} & \alpha \neq d \\ \ln \dfrac{k_{\max}}{k_{\min}} & \alpha = d \end{cases} \tag{B.12}$$

When $\alpha < d$ the numerator is dominated by $k_{\max}$, so high frequencies carry most of the energy. When $\alpha > d$ the term $k_{\min}^{d-\alpha}$ dominates since its exponent is now positive, and low frequencies dominate. The critical case $\alpha = d$ is where the spectral falloff $k^{-\alpha}$ exactly compensates the growing mode count k^{d-1}, producing scale invariance: the energy in any band $[k, \lambda k]$ is

$$E_{[k,\,\lambda k]} \propto \int_{k}^{\lambda k} \frac{dk'}{k'} = \ln \lambda \tag{B.13}$$

which is independent of k. This is also the point where the integral switches from being controlled by its upper limit to its lower limit.

Energy per multiplicative band for general α

More generally, the energy in a band $[k, \lambda k]$ for $\alpha \neq d$ is:

$$E_{[k,\,\lambda k]} = \frac{\Omega_d\, A}{(2\pi)^d} \int_{k}^{\lambda k} k'^{d-1-\alpha}\, dk' = \frac{\Omega_d\, A}{(2\pi)^d} \cdot \frac{\lambda^{d-\alpha} - 1}{d-\alpha}\, k^{d-\alpha} \tag{B.14}$$

The factor $(\lambda^{d-\alpha} - 1)/(d - \alpha)$ is a constant independent of k, so:

$$E_{[k,\,\lambda k]} \propto k^{d-\alpha} \tag{B.15}$$

This is increasing in k when $\alpha < d$ (more energy at high frequencies), constant when $\alpha = d$ (scale invariance), and decreasing when $\alpha > d$ (more energy at low frequencies).

The transition at $\alpha = d$ is the unique point at which the total energy integral switches from being dominated by high frequencies to being dominated by low frequencies. This is a property of the signal itself, independent of any particular filter or processing architecture.

B.2 Supplementary Figures

B.2.1 VGG16 Layer-by-Layer Analysis

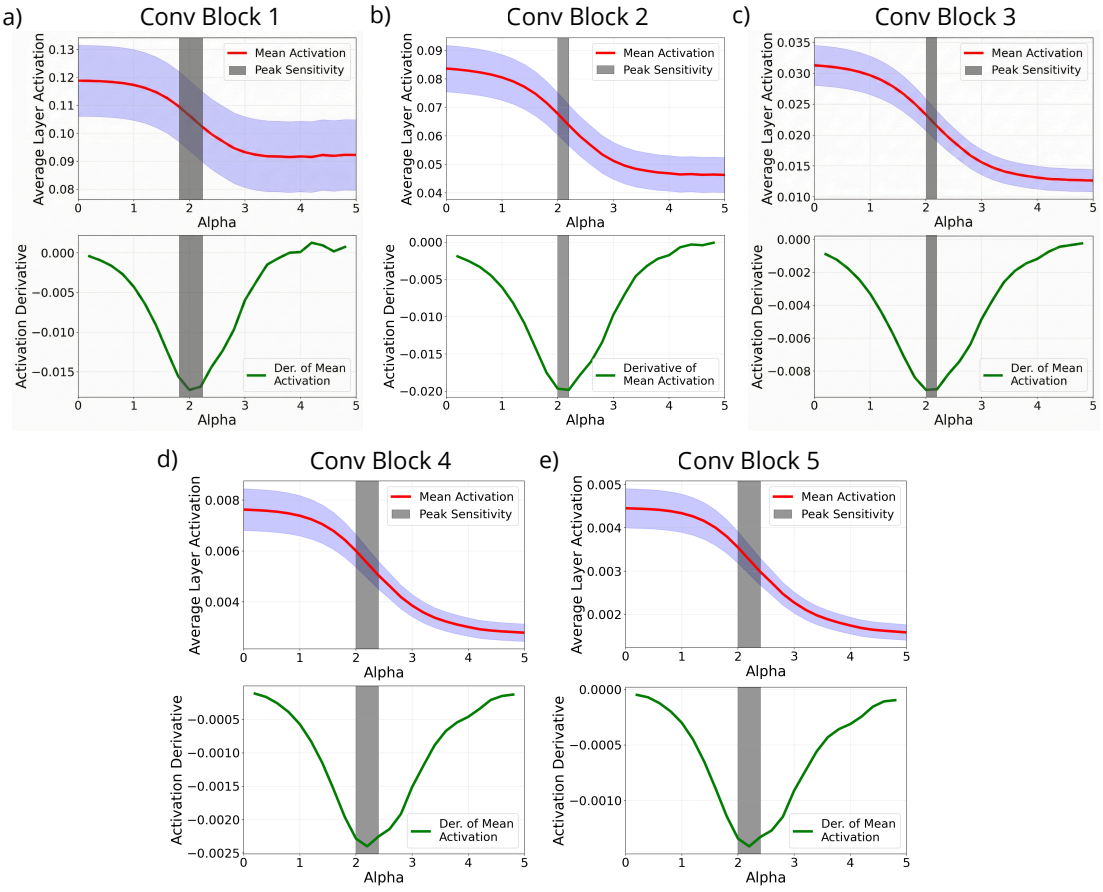

Figure B.2: **Peak sensitivity to natural image statistics holds across all VGG16 convolutional layers.** Each panel (a-e) shows response curves and derivatives for one of VGG16's five convolutional blocks. Top row shows mean activation across all units in each block as a function of α, with red line indicating mean response across 200 images per α value and shaded region showing standard deviation. Bottom row shows the derivative of each activation curve (computed using centered finite differences). Gray bars mark the region of peak sensitivity (most negative derivative). All layers show peak sensitivity near $\alpha \approx 2$, the characteristic slope of natural images, indicating this is a general architectural property rather than layer-specific.

Bibliography

[1] Gordon, A. D. A review of hierarchical classification. *J. R. Stat. Soc. A* **150**, 119–137 (1987).

[2] Aleksander, S. A. *et al.* The gene ontology knowledgebase in 2023. *Genetics* **224**, iyad031 (2023).

[3] Deng, J. *et al.* Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*, 248–255 (IEEE, 2009).

[4] Krizhevsky, A. & Hinton, G. Learning multiple layers of features from tiny images. Tech. Rep. 0, University of Toronto, Toronto, Ontario (2009). URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[5] Esteva, A. *et al.* Dermatologist-level classification of skin cancer with deep neural networks. *Nature* **542**, 115–118 (2017).

[6] Chen, Z., Ding, R., Chin, T.-W. & Marculescu, D. Understanding the impact of label granularity on cnn-based image classification. In *ICDMW*, 895–904 (IEEE, 2018).

[7] Ridnik, T., Ben-Baruch, E., Noy, A. & Zelnik-Manor, L. Imagenet-21k pretraining for the masses. In *NeurIPS DB Track (Round 1)* (PMLR, 2021).

[8] Hong, G. Z., Cui, Y., Fuxman, A., Chan, S. H. & Luo, E. Why fine-grained labels in pretraining benefit generalization? *TMLR* (2024).

[9] Geirhos, R. *et al.* Shortcut learning in deep neural networks. *Nat. Mach. Intell.* **2**, 665–673 (2020).

[10] Tishby, N. & Zaslavsky, N. Deep learning and the information bottleneck principle. In *ITW*, 1–5 (IEEE, 2015).

[11] Achille, A. & Soatto, S. Emergence of invariance and disentanglement in deep representations. *JMLR* **19**, 1–34 (2018).

[12] Ciceri, S. *et al.* Inversion dynamics of class manifolds in deep learning reveals tradeoffs underlying generalization. *Nat. Mach. Intell.* **6**, 40–47 (2024).

[13] Carbonnelle, S. & De Vleeschouwer, C. Intraclass clustering: An implicit learning ability that regularizes DNNs. In *ICLR* (PMLR, 2021).

[14] Hinton, G., Vinyals, O. & Dean, J. Distilling the knowledge in a neural network. *arXiv* (2015).

[15] Müller, R., Kornblith, S. & Hinton, G. E. Subclass distillation. *arXiv* (2020).

[16] Zdeborová, L. Understanding deep learning is also a job for physicists. *Nat. Phys.* **16**, 602–604 (2020).

[17] LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324 (1998).

[18] Clanuwat, T. *et al.* Deep learning for classical japanese literature. *arXiv* (2018).

[19] Xiao, H., Rasul, K. & Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv* (2017).

[20] Krizhevsky, A. & Hinton, G. Learning multiple layers of features from tiny images. Tech. Rep., University of Toronto (2009).

[21] Eldan, R. & Shamir, O. The power of depth for feedforward neural networks. In *COLT*, vol. 49 of *PMLR*, 907–940 (2016).

[22] Suchecki, K., Salah, A. A. A., Gao, C. & Scharnhorst, A. Evolution of wikipedia's category structure. *Adv. Complex Syst.* **15**, 1250068 (2012).

[23] DeGiuli, E. Random language model. *Phys. Rev. Lett.* **122**, 128301 (2019).

[24] Miller, G. A. Wordnet: a lexical database for english. *Commun. ACM* **38**, 39–41 (1995).

[25] Poggio, T., Banburski, A. & Liao, Q. Theoretical issues in deep networks. *PNAS* **117**, 30039–30045 (2020).

[26] Danhofer, D. A., D'Ascenzo, D., Dubach, R. & Poggio, T. Position: A theory of deep learning must include compositional sparsity. In *ICML* (PMLR, 2025).

[27] Yamins, D. L. K. *et al.* Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences* **111**, 8619–8624 (2014).

[28] Yamins, D. L. K. & DiCarlo, J. J. Using goal-driven deep learning models to understand sensory cortex. *Nature Neuroscience* **19**, 356–365 (2016).

[29] Güçlü, U. & van Gerven, M. A. J. Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream. *Journal of Neuroscience* **35**, 10005–10014 (2015).

[30] Khaligh-Razavi, S.-M. & Kriegeskorte, N. Deep supervised, but not unsupervised, models may explain it cortical representation. *PLOS Computational Biology* **10**, 1–29 (2014). URL https://doi.org/10.1371/journal.pcbi.1003915.

[31] Field, D. J. Relations between the statistics of natural images and the response properties of cortical cells. *J. Opt. Soc. Am. A* **4**, 2379–2394 (1987). URL https://opg.optica.org/josaa/abstract.cfm?URI=josaa-4-12-2379.

[32] Ruderman, D. L. & Bialek, W. Statistics of natural images: Scaling in the woods. *Phys. Rev. Lett.* **73**, 814–817 (1994). URL https://link.aps.org/doi/10.1103/PhysRevLett.73.814.

[33] Atick, J. J. Entropy minimization: a design principle for sensory perception? *Int. J. Neural Syst.* **3**, 81–90 (1992). URL https://doi.org/10.1142/S0129065792000413.

[34] Pitkow, X. & Meister, M. Decorrelation and efficient coding by retinal ganglion cells. *Nature Neuroscience* **15**, 628–635 (2012).

[35] Mahuas, G., Isacchini, G., Marre, O., Ferrari, U. & Mora, T. A new inference approach for training shallow and deep generalized linear models of noisy interacting neurons. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. & Lin, H. (eds.) *Advances in Neural Information Processing Systems*, vol. 33, 5070–5080 (Curran Associates, Inc., 2020). URL https://proceedings.neurips.cc/paper_files/paper/2020/file/356dc40642abeb3a437e7e06f178701c-Paper.pdf.

[36] Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition (2015). URL https://arxiv.org/abs/1409.1556. 1409.1556.

[37] Torralba, A. & Oliva, A. Statistics of natural image categories. *Network: Computation in Neural Systems* **14**, 391–412 (2003).

[38] Mahuas, G., Buffet, T., Marre, O., Ferrari, U. & Mora, T. Strong, but not weak, noise correlations are beneficial for population coding. *PRX Life* **3**, 033012 (2025). URL https://link.aps.org/doi/10.1103/nhvz-6grt.

[39] He, K., Zhang, X., Ren, S. & Sun, J. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification 1026–1034 (2015).

[40] van der Maaten, L. & Hinton, G. Visualizing data using t-sne. *Journal of Machine Learning Research* **9**, 2579–2605 (2008).