Comparative Generalization Bounds for Deep Neural Networks

Anonymous authors
Paper under double-blind review

Abstract

In this work, we investigate the generalization capabilities of deep neural networks. We introduce a measure of the effective depth of neural networks, defined as the first layer at which sample embeddings are separable using the nearest-class center classifier. Our empirical results demonstrate that, in standard classification settings, neural networks trained using Stochastic Gradient Descent tend to have small effective depths. We also explore the relationship between effective depth, the complexity of the training dataset, and generalization. For instance, we find that the effective depth of a trained neural network increases as the number of random labels in the data increases. Additionally, we derive a generalization bound by comparing the effective depth of a network with the minimal depth required to fit the same dataset with partially corrupted labels. This bound provides non-vacuous predictions of test performance and is found to be independent of the actual depth of the network in our experiments.

1 Introduction

Deep learning systems have steadily advanced the state of the art in a wide range of benchmarks, demonstrating impressive performance in tasks ranging from image classification (Taigman et al., 2014; Zhai et al., 2021), language processing (Devlin et al., 2019; Brown et al., 2020), open-ended environments (Silver et al., 2016; Arulkumaran et al., 2019), to coding (Chen et al., 2021).

Recent research suggests that deep neural networks are able to generalize well to new data because they have a large number of parameters relative to the number of training samples (Belkin et al., 2018; Belkin, 2021; Advani & Saxe, 2017; Belkin et al., 2019). However, it has been shown that in these cases deep learning models can also precisely interpolate arbitrary training labels (Zhang et al., 2017), a phenomenon known as the "interpolation regime." Understanding how deep learning models learn through interpolation is an important step towards a more comprehensive theoretical understanding of their successes.

Traditional generalization bounds (Vapnik, 1998; Shalev-Shwartz & Ben-David, 2014; Mohri et al., 2012; Bartlett & Mendelson, 2003) are based on uniform convergence and are used to control the worst-case generalization gap (the difference between train and test errors) over a set of predictors that includes the outputs of a learning algorithm. However, the applicability of these bounds to certain interpolation learning regimes has been called into question by Nagarajan & Kolter (2019), who described theoretical scenarios where an interpolation learning algorithm generalizes well but a uniform convergence bound cannot detect this. Subsequent research by Bartlett & Long (2021); Zhou et al. (2020); Negrea et al. (2020); Yang et al. (2021) has also demonstrated the limitations of uniform convergence in various interpolation learning situations.

Contributions. In this paper, we present a novel approach for measuring generalization in deep learning that does not rely on uniform convergence bounds. Instead, our bound suggests that a model will perform well at test time if its complexity is small compared to the complexity of a network required to fit the same dataset with partially random labels. In other words, even if a trained network has a complexity greater than the number of training samples, it may still be less complex than a model that fits partially random labels. As a result, in such cases, our bound may provide a non-trivial estimate of the test error.

Dataset Architecture	MNIST CONV-L-50			Fashion MNIST CONV-L-100			CIFAR10 CONV-L-100			CIFAR10 CONVRES-L-50		
Depth (L)	10	12	15	10	12	15	16	18	20	10	12	15
Test error	0.0075	0.0074	0.0074	0.0996	0.0996	0.0996	0.2659	0.2653	0.2648	0.2903	0.2862	0.2804
p	0.1	0.1	0.1	0.2	0.2	0.2	0.4	0.4	0.4	0.4	0.4	0.4
Our bound	**0.1**	**0.1**	**0.1**	**0.2**	**0.2**	**0.2**	**0.66**	**0.66**	**0.53**	**0.4**	**0.4**	**0.4**
$L_{1,\infty}$ (Bartlett & Mendelson, 2003)	8.911e+14	1.74e+17	2.13e+22	3.613e+17	9.145e+18	4.088e+22	1.076e+23	6.682e+28	2.758e+35	-	-	-
$L_{3,1.5}$ (Neyshabur et al., 2015)	5.462e+05	1.6e+06	1.308e+06	7.523e+07	6.997e+07	2.636e+08	4.633e+08	2.275e+09	5.061e+09	-	-	-
Frobenius (Neyshabur et al., 2015)	1.848e+06	8.194e+06	2.216e+07	2.486e+08	2.335e+08	1.585e+09	1.967e+09	1.442e+10	3.038e+11	-	-	-
Spec L_1 (Bartlett et al., 2017)	2.861e+05	6.412e+05	9.566e+05	4.706e+06	3.516e+06	3.176e+06	1.19e+07	1.449e+08	1.272e+10	-	-	-
Spec Frob (Neyshabur et al., 2019)	3.948e+03	1.1199e+04	1.538e+04	4.0229e+04	2.884e+04	2.543e+04	9.4833e+04	1.011e+06	1.033e+08	-	-	-

Table 1: **Comparing our bound with baseline bounds in the literature for networks of varying depths.** Our error bound is reported in the fourth row, and the baseline bounds are reported in the bottom rectangle. While the test error is universally bounded by 1, the baseline bounds are much larger than 1, and therefore, are meaningless. **In contrast, our bound achieves relatively tight estimations of the test error and unlike the baseline bounds, our bound is fairly unaffected by the network's depth.**

To formally describe our notion of complexity, we use the concept of nearest class-center (NCC) separability. This property states that the feature embeddings associated with training samples belonging to the same class can be separated using the nearest class-center decision rule. While earlier research (Papyan et al., 2020) found that NCC separability occurs at the penultimate layer of trained networks, more recent research (Ben-Shaul & Dekel, 2022) has discovered NCC separability in intermediate layers as well. In this work, we introduce the concept of "effective depth" in neural networks, which refers to the lowest layer at which the features are NCC separable (see Sec. 3.2).

We have made several key observations about effective depths. First, we have found that the effective depth of trained networks increases as the amount of random labels in the data increases. Second, when training deep networks, we have observed that they tend to converge to an effective depth L_0, regardless of their actual depth L. This means that the feature embeddings of layers above L_0 tend to be NCC separable. In addition, we have shown in Tab. 1 that our bound on generalization is empirical, non-vacuous, and independent of depth, unlike traditional bounds. In Section 3.3 we further discuss the limitations of modern norm-based generalization bounds (e.g., Neyshabur et al. (2015); Bartlett et al. (2017); Golowich et al. (2017); Neyshabur et al. (2018)), along with the key distinctions between these bounds and the proposed bound.

1.1 Additional Related Work

There has been significant research on the geometrical properties of intermediate layers in deep neural networks, such as clustering and separability (Papyan, 2020; Tirer & Bruna, 2022; Galanti et al., 2022; Ben-Shaul & Dekel, 2022; Cohen et al., 2018; Alain & Bengio, 2017; Montavon et al., 2011; Papyan et al., 2017; Ben-Shaul & Dekel, 2021; Shwartz-Ziv & Tishby, 2017). While previous studies have analyzed these properties theoretically (Zhu et al., 2021; Rangamani et al., 2022; Lu & Steinerberger, 2020; Fang et al., 2021; Ergen & Pilanci, 2021), their specific role in deep learning and potential relationship with generalization are not yet fully understood. We focus on the question of whether these properties are good indicator of generalization. In contrast, previous research (Zhu et al., 2021) has shown that such properties may occur even when training a network with random labels, suggesting that they may not directly indicate generalization. In this paper, we argue that effective depth can be used to measure the complexity of fitting a dataset, and show how this idea can help us predict test performance.

2 Problem Setup

In this section, we explain the learning setting used in our theory and experiments. We focus on the task of training a model for standard multi-class classification. Specifically, we consider a distribution P over samples (x, y) where x belongs to the instance space $\mathcal{X}$, and y belongs to the label space $\mathcal{Y}_C$ with a cardinality of C. To simplify, we use one-hot encoding for the label space, where labels are represented by unit vectors in $\mathbb{R}^C$, and $\mathcal{Y}_C = \{e_c \mid c = 1, \dots, C\}$ and e_c is the cth standard unit vector in $\mathbb{R}^C$. We also use the notation $y = c$ instead of $y = e_c$. The class conditional distribution of x given $y = c$ is denoted as $P_c(\cdot) := \mathbb{P}[x \in \cdot \mid y = c]$.

A classifier $h_W : \mathcal{X} \to \mathbb{R}^C$ assigns a *soft* label to an input point $x \in \mathcal{X}$, and its performance on the distribution P is measured by the expected risk

$$L_P(h_W) := \mathbb{E}_{(x,y(x))\sim P}[\ell(h_W(x), y(x))],$$

where $\ell : \mathbb{R}^C \times \mathcal{Y}_C \to [0, \infty)$ is a non-negative loss function (e.g., L_2 or cross-entropy losses).

We typically do not have direct access to the full population distribution P. Therefore, we generally aim to learn a classifier, h, using some balanced training data $S := \{(x_i, y_i)\}_{i=1}^m = \cup_{c=1}^C S_c = \cup_{c=1}^C \{x_{ci}, y_{ci}\}_{i=1}^{m_0} \sim P_B(m)$ of $m = C \cdot m_0$ samples consisting m_0 independent and identically distributed (i.i.d.) samples drawn from P_c for each $c \in [C]$. Specifically, we intend to find W that minimizes the regularized empirical risk

$$L_S^\lambda(h_W) \;:=\; \frac{1}{m}\sum_{i=1}^m \ell(h_W(x_i), y_i) + \lambda\|W\|_2^2, \tag{1}$$

where the regularization controls the complexity of the function h_W and typically helps reducing overfitting. Finally, the performance of the trained model is evaluated using the train and test error rates; $\text{err}_S(h_W) := \frac{1}{m}\sum_{i=1}^m \mathbb{I}[\arg\max_c h_W(x_i)_c \neq y_i]$ and $\text{err}_P(h_W) := \mathbb{E}_{(x,y)\sim P}[\mathbb{I}[\arg\max_c h_W(x)_c \neq y]]$, where $\mathbb{I} : \{\text{True}, \text{False}\} \to \{0, 1\}$ the indicator function.

Neural networks. In this work, the classifier h_W is a neural network composed of a set of parametric layers. It is written as $h_W := e_{W_e} \circ f_{W_f}^L := e_{W_e} \circ g_{W_L}^L \circ \cdots \circ g_{W_1}^1$, where $g_{W_i}^i$ are parametric functions that map from $\mathbb{R}^{p_i}$ to $\mathbb{R}^{p_{i+1}}$, and e_{W_e} is a linear function that maps from $\mathbb{R}^{p_{L+1}}$ to $\mathbb{R}^C$. These layers can be standard linear or convolutional layers (with ReLU activations) or a residual block. To simplify notation, we denote $f_i := g^i \circ \cdots \circ g^1$ and $h := h_W$. The specific architectures used in the experiments are described Appendix A.1.

Optimization. We optimize our models by minimizing the regularized empirical risk $L_S^\lambda(h)$ using Stochastic Gradient Descent (SGD) for a certain number of iterations T with a regularization coefficient $\lambda > 0$. To do this, we initialize the weights $W_0 = \gamma$ of h with a standard initialization procedure and at each iteration, update $W_{t+1} \leftarrow W_t - \mu_t \nabla_W L_{\tilde{S}}(h_t)$, where $\mu_t > 0$ is the learning rate at the t-th iteration, and $\tilde{S} \subset S$ is a subset of size B selected uniformly at random. Throughout the paper, we denote by h_S^γ the output of the learning algorithm starting from initialization $W_0 = \gamma$. When γ is not relevant or is obvious from the context, we will simply write $h_S^\gamma = h_S = e_S \circ f_S$.

3 Neural Collapse and Generalization

In this section, we examine the theoretical connection between neural collapse and generalization. We begin by defining neural collapse, NCC separability, and effective depth of neural networks. We then explore how these concepts relate to the test-time performance of neural networks.

3.1 Nearest Class-Center Separability

Neural collapse (Papyan et al., 2020) identifies training dynamics of deep networks for standard classification tasks, in which the features of the penultimate layer associated with training samples belonging to the same class tend to concentrate around their class-means. This includes (NC1) class-features variability collapse, (NC2) the class means of the embeddings collapse to the vertices of a simplex equiangular tight frame, (NC3) the last-layer classifiers collapse to the class means up to scaling and (NC4) the classifier's decision collapses to simply choosing whichever class has the closest train class mean, while maintaining a zero classification error.

In this paper we focus on a weak form of NC4 we call *"nearest class-center separability"* (NCC separability). Formally, suppose we have a dataset $S = \cup_{c=1}^C S_c$ of samples and a mapping $f : \mathbb{R}^d \to \mathbb{R}^p$, the features of f are NCC separable (w.r.t. S) if for all $i \in [m]$, we have $\hat{h}(x_i) = y_i$, where

$$\hat{h}(x) \;:=\; \underset{c\in[C]}{\arg\min}\, \|f(x) - \mu_f(S_c)\|. \tag{2}$$

To measure the degree of NCC separability of a feature map f, we use the train and test classification error rates of the NCC classifier on top of the given layer, $\mathrm{err}_S(\hat{h})$ and $\mathrm{err}_P(\hat{h})$.

Essentially, NC4 asserts that during training, the feature embeddings in the penultimate layer become separable and the classifier h itself converges to the 'nearest class-center classifier' $\hat{h}$.

3.2 Effective Depths and Generalization

In this section we study the effective depths of neural networks and their connection with generalization. To formally define this notion, we focus on neural networks whose L top-most layers are of the same size. We observe that neural networks trained for standard classification exhibit an implicit bias towards depth minimization.

Observation 1 (Minimal depth hypothesis)**.** *Suppose we have a dataset S. There exists an integer $L_0 \geq 1$, such that, if we train a neural network of any depth $L \geq L_0$ for cross-entropy minimization on S using SGD with weight decay, the learned features f^l become (approximately) NCC separable for all $l \in \{L_0, \dots, L\}$.*

We note that if the L_0'th layer of f_L exhibits NCC separability, we could correctly classify the samples already in the L_0'th layer of f_L using a linear classifier (i.e., the nearest class-center classifier). Therefore, intuitively its depth is effectively upper bounded by L_0. The notion of effective depth of a neural network is formally defined as follows.

Definition 1 (ϵ-effective depth)**.** *Suppose we have a dataset S and a neural network $h = e \circ g^L \circ \cdots \circ g^1$ with $g^1 : \mathbb{R}^n \to \mathbb{R}^{p_2}$, $g^i : \mathbb{R}^{p_i} \to \mathbb{R}^{p_{i+1}}$ and linear classifier $e : \mathbb{R}^{p_{L+1}} \to \mathbb{R}^C$. Let $\hat{h}_i(x) := \arg\min_{c \in [C]} \|f_i(x) - \mu_{f_i}(S_c)\|$. The ϵ-effective depth $\mathscr{d}^{\epsilon}_S(h)$ of the network h is the minimal value $i \in [L]$, such that, $\mathrm{err}_S(\hat{h}_i) \leq \epsilon$ (and $\mathscr{d}^{\epsilon}_S(h) = L$ if such $i \in [L]$ is non-existent).*

To avoid confusion, we note that the ϵ-effective depth is a property of a neural network and not of the function it implements. That is, a function can be implemented by two different architectures of different effective depths. While our empirical observations in Sec. 4 suggest that the optimizer learns neural networks of low-depths, it is not necessarily the lowest depth that allows NCC separability. As a next step, we define the *ϵ-minimal NCC depth.* Intuitively, the NCC depth of a given architecture is the minimal value $L \in \mathbb{N}$, for which there exists a neural network of depth L whose features are NCC separable. As we will show, the relationship between the ϵ-effective depth of a neural network and the ϵ-minimal NCC depth is connected with generalization.

Definition 2 (ϵ-Minimal NCC depth)**.** *Suppose we have a dataset $S = \cup_{c=1}^{C} S_c$ and a neural network architecture $f^L = g^L \circ \cdots \circ g^1$ with $g^1 : \mathbb{R}^n \to \mathbb{R}^{n_0}$ and $g^i \in \mathcal{G} \subset \{g' \mid g' : \mathbb{R}^{n_0} \to \mathbb{R}^{n_0}\}$ for all $i = 2, \dots, L$. The ϵ-minimal NCC depth of $\mathcal{G}$ is the minimal depth L for which there exist parameters $W = \{W_i\}_{i=1}^{L}$, such that, $f' := f^L_W = g^L_{W_L} \circ \cdots \circ g^1_{W_1}$ satisfies $\mathrm{err}_S(\hat{h}) \leq \epsilon$, where $\hat{h}(x) := \arg\min_{c \in [C]} \|f'(x) - \mu_{f'}(S_c)\|$. We denote the ϵ-minimal NCC depth by $\mathscr{d}^{\epsilon}_{\min}(\mathcal{G}, S)$.*

To study the performance of a given model, we consider the following setup. Let $S_1 = \{(x^1_i, y^1_i)\}_{i=1}^{m}$ and $S_2 = \{(x^2_i, y^2_i)\}_{i=1}^{m}$ be two balanced datasets. We think of them as two splits of the training dataset S. We assume that the classifier $h^{\gamma}_{S_1}$ is trained on S_1 and we use S_2 to evaluate its performance. We denote by $X_j = \{x^j_i\}_{i=1}^{m}$ and $Y_j = \{y^j_i\}_{i=1}^{m}$ the instances and labels in S_j.

To formally state our bound, we make two technical assumptions. The first is that the misclassified labels that $h^{\gamma}_{S_1}$ produces over the samples $X_2 = \cup_{c=1}^{C} \{x^2_{ci}\}_{i=1}^{m_0}$ are distributed uniformly.

Definition 3 (δ_m-uniform mistakes)**.** *We say that the mistakes of a learning algorithm $A : (S_1, \gamma) \mapsto h^{\gamma}_{S_1}$ are δ_m-uniform, if with probability $\geq 1 - \delta_m$ over the selection of $S_1, S_2 \sim P_B(m)$, the values and indices of the mistaken labels of $h^{\gamma}_{S_1}$ over X_2 are uniformly distributed (as a function of γ).*

The above definition provides two conditions regarding the learning algorithm. It assumes that with a high probability (over the selection of S_1, S_2), $h^{\gamma}_{S_1}$ makes the same number of mistakes on S_2 across all initializations γ. In addition, it assumes that the mistakes are distributed uniformly across the samples in S_2 and their (incorrect) values are also distributed uniformly. While these assumptions may be violated in practice, the train error typically has a small variance and the mistakes are almost distributed uniformly when the classes are non-hierarchical (e.g., CIFAR10, MNIST).

For the second assumption, we consider the following term. Let $p \in (0, 1/2), \alpha \in (0, 1)$, we denote

$$\delta^2_{m,p,\alpha} \;:=\; \mathbb{P}_{S_1,S_2,\tilde{Y}_2,\hat{Y}_2}\left[\exists\, q \geq (1+\alpha)\, p :\; \mathscr{d}^{\epsilon}_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2) > \mathbb{E}_{\hat{Y}_2}[\mathscr{d}^{\epsilon}_{\min}(\mathcal{G}, S_1 \cup \hat{S}_2)]\right], \tag{3}$$

where $\tilde{Y}_2 = \{\tilde{y}_i\}_{i=1}^m$ and $\hat{Y}_2 = \{\hat{y}_i\}_{i=1}^m$ are uniformly selected to be sets of labels that disagree with Y_2 on pm and qm values (resp.) and $\tilde{S}_2$ and $\hat{S}_2$ are datasets obtained by replacing the labels of S_2 with $\tilde{Y}_2$ and $\hat{Y}_2$ (resp.). We assume that $\delta^2_{m,p,\alpha}$ is small. Meaning, with a high probability, the minimal depth to fit $(2-p)m$ correct labels and pm random labels is upper bounded by the expected minimal depth to fit $(2-q)m$ correct labels and qm random labels for any $q \geq (1+\alpha)p$. To understand this assumption, we note that in both cases, the model has to fit at least m correct labels and pm (or qm) random labels. However, we typically need to increase the capacity of the model in order to fit extended amounts of random labels (see Figs. 3).

Following the setting above, we are prepared to formulate our generalization bound.

Proposition 1. *Let $m \in \mathbb{N}$, $p \in (0, 1/2)$, $\alpha \in (0, 1)$ and $\epsilon \in (0, 1)$. Assume that the error of the learning algorithm is δ^1_m-uniform. Assume that $S_1, S_2 \sim P_B(m)$. Let $h^{\gamma}_{S_1}$ be the output of the learning algorithm given access to a dataset S_1 and initialization γ. Then,*

$$\begin{aligned}\mathbb{E}_{S_1}\mathbb{E}_{\gamma}[\mathrm{err}_P(h^{\gamma}_{S_1})] \;\leq\; & \mathbb{P}_{S_1,S_2,\tilde{Y}_2}\left[\mathbb{E}_{\gamma}[\mathscr{d}^{\epsilon}_{S_1}(h^{\gamma}_{S_1})] \;\geq\; \mathscr{d}^{\epsilon}_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2)\right] \\ & + (1+\alpha)p + \delta^1_m + \delta^2_{m,p,\alpha},\end{aligned} \tag{4}$$

where $\tilde{Y}_2 = \{\tilde{y}_i\}_{i=1}^m$ is uniformly selected to be a set of labels that disagrees with Y_2 on pm values.

The above proposition provides an upper bound on the expected test error of the classifier $h^{\gamma}_{S_1}$ which is the term that we would like to bound. The proposition assumes that the mistakes $h^{\gamma}_{S_1}$ generates on X_2 are distributed uniformly (with probability $\geq 1 - \delta^1_m$). To account the likelihood that this assumption fails, our bound includes the term δ^1_m, which is assumed to be small.

Informally, the bound suggests the following idea to evaluate the performance of $h^{\gamma}_{S_1}$. We start with an initial guess $p_m = p \in (0, 1/2)$ of the test error of $h^{\gamma}_{S_1}$. Using this guess, we compare its ϵ-effective depth with the ϵ-minimal NCC depth $\mathscr{d}^{\epsilon}_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2)$ required to NCC separate the samples in $S_1 \cup \tilde{S}_2$, where $\tilde{S}_2$ is the result of randomly relabeling $p_m m$ of S_2's labels. Intuitively, if the mistakes of $h^{\gamma}_{S_1}$ are uniformly distributed and its ϵ-effective depth is smaller than $\mathscr{d}^{\epsilon}_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2)$, then, we expect $h^{\gamma}_{S_1}$ to make at most p_m mistakes on S_2. Therefore, in a sense, the choice of p_m serves as a 'guess' whether the effective depth of a model trained with S_1 is likely to be smaller than the ϵ-minimal NCC depth required to NCC separate the samples in $S_1 \cup \tilde{S}_2$.

Next, we interpret each term separately. The term $\mathbb{E}_{\gamma}[\mathscr{d}^{\epsilon}_{S_1}(h^{\gamma}_{S_1})]$ depends on the complexity of the classification problem and the implicit bias of SGD to favor networks of small ϵ-effective depths. In the worst case, if SGD does not minimize the ϵ-effective depth or the labels in S_1 are random (and m is sufficiently large), we expect $\mathbb{E}_{\gamma}[\mathscr{d}^{\epsilon}_{S_1}(h^{\gamma}_{S_1})] = L$. On the other hand, $\mathscr{d}^{\epsilon}_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2)$ measures the complexity of a task that involves fitting a dataset of size $2m$ samples, where $(2 - p_m)m \geq m$ of the labels are correct and $p_m m$ are random labels. By decreasing p_m, we expect $\mathscr{d}^{\epsilon}_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2)$ to decrease, making the first term in the bound larger. In addition, if $h = e \circ f^L$ is a neural network of a fixed width, it is impossible to fit an increasing amount of random labels without increasing the depth. Therefore, when $p_m m \underset{m\to\infty}{\longrightarrow} \infty$, the dataset $S_1 \cup \tilde{S}_2$ becomes increasingly harder to fit, and we expect $\mathscr{d}^{\epsilon}_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2)$ to tend to infinity. If $\mathbb{E}_{\gamma}[\mathscr{d}^{\epsilon}_{S_1}(h^{\gamma}_{S_1})]$ is bounded as a function of L and m and if $p_m m \underset{m\to\infty}{\longrightarrow} \infty$, we obtain that $\mathbb{P}[\mathbb{E}_{\gamma}[\mathscr{d}^{\epsilon}_{S_1}(h^{\gamma}_{S_1})] \geq \mathscr{d}^{\epsilon}_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2)] \underset{m\to\infty}{\longrightarrow} 0$ and together with $p_m \underset{m\to\infty}{\longrightarrow} 0$, we have $\mathbb{E}_{S_1}[\mathrm{err}_P(h_{S_1})] \leq \delta^1_m + \delta^2_{m,p,\alpha} + o_m(1)$.

As a side note, computing the expectation over S_1, S_2 in the bound is impossible, due to the limited access of the training data. However, instead, we empirically estimate this term using a set of k pairs (S^i_1, S^i_2) of m samples, yielding an additional term that scales as $\mathcal{O}(1/\sqrt{k})$ to the bound (see Prop. 2 in the appendix).

3.3 Comparing Prop. 1 with Standard Generalization Bounds

Classic bounds (e.g., (Vapnik, 1998)) are based on bounding the test error with the sum between the train error together with a term $\mathcal{O}(\sqrt{\mathcal{C}(\mathcal{H})/m})$, where $\mathcal{C}(\mathcal{H})$ measures the complexity (e.g., VC dimension) of the class $\mathcal{H}$ (e.g., neural networks) and m is the number of training samples. However, as discussed in Sec. 1, these bounds are vacuous in overparameterized learning regimes (e.g., training ResNet-50 on CIFAR10 classification). For instance, for VC-dimension based bounds (Vapnik, 1998), $\mathcal{C}(\mathcal{H})$ equals the VC-dimension of the class $\mathcal{H}$ which scales with the number of trainable parameters for ReLU networks (Bartlett et al., 2019). For example, even though the ResNet-50 architecture generalizes well when trained on CIFAR10, it has over 23 million parameters compared to the $m = 50000$ training samples in the dataset.

More recently, Neyshabur et al. (2015); Bartlett et al. (2017); Golowich et al. (2017); Neyshabur et al. (2018) suggested generalization bounds for neural networks that weakly depend on uniform convergence. In these bounds, the class-complexity $\mathcal{C}(\mathcal{H})$ is replaced with the individual complexity $\mathcal{C}(h_W)$ of the function we learn. For example, Golowich et al. (2017) proposed bounds that scale with $\mathcal{C}(h_W) = \rho^2 L$, where L is the depth of h_W and ρ measures the product of the norms of its weight matrices. However, Nagarajan & Kolter (2019) showed that in certain cases unregularized least squares can generalize well even when its norm ρ scales as $\Theta(\sqrt{m})$ and the bound becomes $\Theta_m(1)$. Furthermore, these bounds tend to be very large in practice (see Tab. 8 in (Neyshabur et al., 2019) and Tab. 1) and are negatively correlated with the test performance (Jiang et al., 2020). In addition, if the network's weight matrices' norms are larger than 1, quantities like ρ grow exponentially when L is varied. As shown in Tab. 1 this is empirically the case.

Our Prop. 1 offers a different way to measure generalization. Since this bound is not based on uniform convergence, it does not require that the network's complexity would be small in comparison to m; rather, the bound guarantees generalization if the network's effective size is smaller than that of a network that fits partially random labels. For instance, when the optimizer has a strong bias towards minimizing the effective depth, $\mathbb{E}_\gamma[d^\epsilon_{S_1}(h^\gamma_{S_1})] \approx d^\epsilon_{\min}(\mathcal{G}, S_1)$ which is by definition upper bounded by $d^\epsilon_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2)$. We note that $d^\epsilon_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2)$ grows to infinity as $m \to \infty$ (since the network needs to memorize $m \to \infty$ random labels). On the other hand, $d^\epsilon_{\min}(\mathcal{G}, S_1)$ is bounded by the depth of a network that approximates the target function y up to an approximation error ϵ (which typically exists due to universal approximation arguments). Therefore, for sufficiently large m, we expect to have $d^\epsilon_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2) > d^\epsilon_{\min}(\mathcal{G}, S_1)$. As we empirically see in Sec. 4, the effective depths of SGD-trained networks are usually small.

Unlike previous bounds, our bound has the advantage of being fairly independent of L. Namely, when the minimal depth hypothesis (Obs. 1) holds, we expect $\mathbb{E}_\gamma[d^\epsilon_{S_1}(h^\gamma_{S_1})]$ to be unaffected by the depth L of $h^\gamma_{S_1}$ (as long as $L \geq L_0$). Since $d^\epsilon_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2)$ is by definition independent of L, we expect $\mathbb{P}[\mathbb{E}_\gamma[d^\epsilon_{S_1}(h^\gamma_{S_1})] \geq d^\epsilon_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2)]$ to be independent of L (when $L \geq L_0$). In Tab. 1 we empirically validate that our bound does not grow when increasing L.

4 Experiments

In this section, we experimentally analyze the emergence of neural collapse in the intermediate layers of neural networks. First, we validate the "Minimal Depth Hypothesis" (Obs. 1). Following that, we look at how corrupted labels affect the extent of intermediate layer NCC separability and the ϵ-effective depth. We show that as the number of corrupted labels in the data increases, so does the ϵ-effective depth. Finally, using the bound in Prop. 1, we provide non-trivial estimates of the test error. In Tab. 1, we empirically compare our bound with relevant baselines and show that, unlike other bounds, it achieves non-vacuous estimations of the test error. Throughout the experiments, we used Tesla-k80 GPUs for several hundred runs. Each run took between 5-20 hours. For additional experiments, see Appendix A. The plots are high-definition pictures and are best viewed when zoomed in.

4.1 Setup

Training process. We consider k-class classification problems (e.g., CIFAR10) and train multilayered neural networks $h = e \circ f^L = e \circ g^L \circ \cdots \circ g^1 : \mathbb{R}^n \to \mathbb{R}^C$ on the corresponding training dataset S. The models

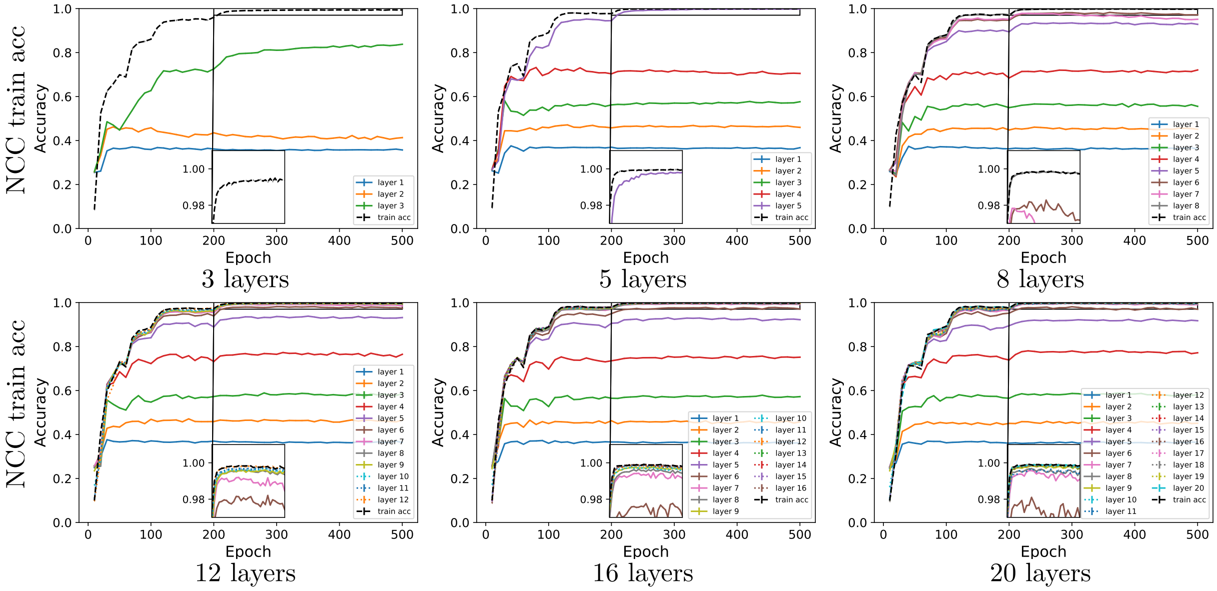


Figure 1: **Intermediate NCC separability of CONV-L-400 trained on CIFAR10.** We plot the NCC train accuracy rates of neural networks with varying numbers of layers. Each curve stands for a different layer within the network.

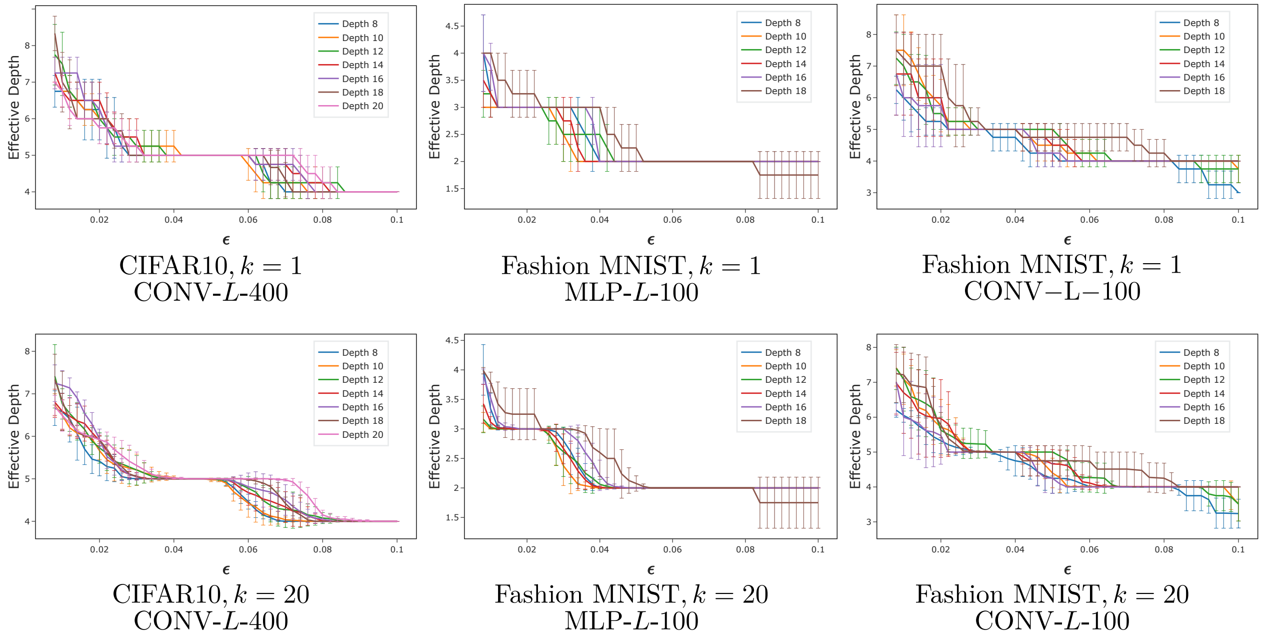


Figure 2: **Averaged ϵ-effective depths over the last few epochs.** We plot the ϵ-effective depth (y-axis) as a function of ϵ (x-axis). Each line specifies the ϵ-effective depth of a neural network of a certain depth L. We show the averaged ϵ-effective depth over the last $k = 1, 20$ epochs across 5 initializations. The network's architecture, dataset and k are specified below each plot.

are trained with SGD for cross-entropy loss minimization between its logits and the one-hot encodings of the labels. We consistently use batch size 128, learning rate schedule with an initial learning rate 0.1, decayed three times by a factor of 0.1 at epochs 60, 120, and 160, momentum 0.9 and weight decay 5e−4. Each model is trained for 500 epochs.

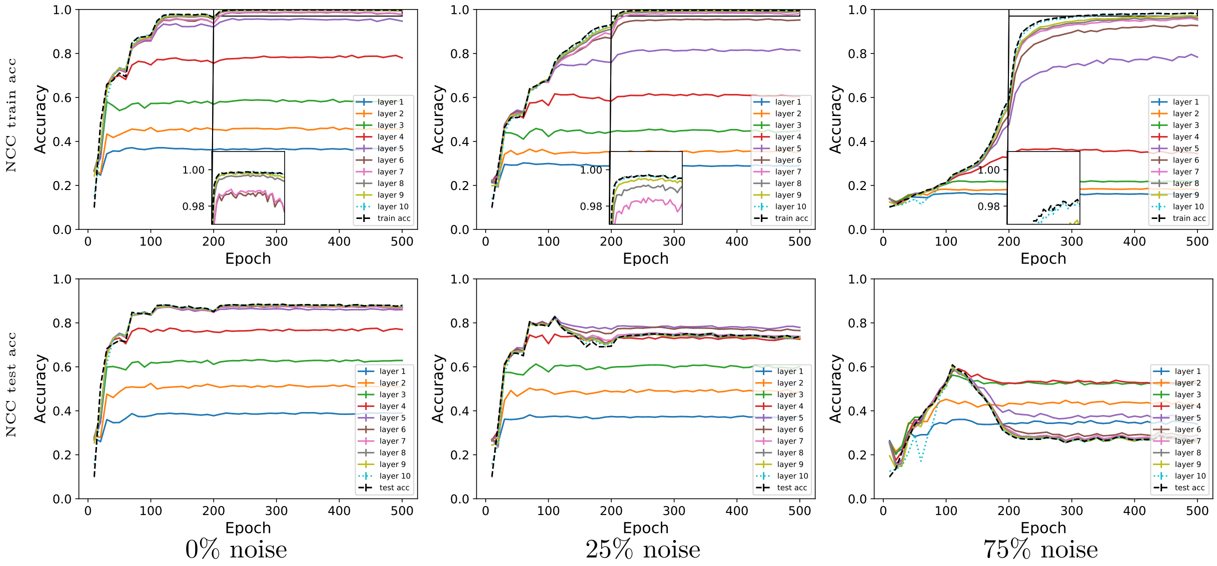


Figure 3: **Intermediate NCC separability of CONV-10-400 trained on CIFAR10 with partially corrupted labels.** We plot the NCC train/test accuracy rates of the various layers of a network trained with a certain amount of corrupted labels (see titles).

Dataset	MNIST			Fashion MNIST			CIFAR10			CIFAR10		
Architecture	CONV-10-50			CONV-10-100			CONV-16-100			CONVRES-10-50		
$\mathbb{E}_{S_1,\gamma}[\text{err}_P(h^{\gamma}_{S_1})]$	0.0075			0.0996			0.2676			0.29		
p	0.05	0.075	0.1	0.05	0.15	0.2	0.4	0.45	0.5	0.1	0.4	0.5
Bound	1.05	0.475	**0.1**	1.05	0.75	**0.2**	**0.66**	0.72	0.7	**0.4**	**0.4**	0.5

Table 2: Estimating the bound in Prop. 1. We used $\epsilon = 0.005$ to measure the effective depths.

Architectures. We focused on three types of architectures: (a) MLP-L-H with L fully-connected layers of width H, (b) CONV-L-H with L 3×3 convolutional layers with padding 1, stride 1 and H output channels and (c) a residual convolutional network CONVRES-L-H with L residual blocks with two 3×3 convolutional layers. In each network the layers are interlaced with batch normalization layers and ReLU activations. For more details see Appendix A.1.

Datasets. We consider various datasets: MNIST, Fashion MNIST, and CIFAR10. For CIFAR10 we used random cropping, random horizontal flips, and random rotations (by $15k$ degrees for k uniformly sampled from [24]). All datasets were standardized.

4.2 Results

Intermediate neural collapse. To investigate the bias towards depth minimization, we trained several CONV-L-400 networks with varying depths on CIFAR10. Each plot in Fig. 1 illustrates the train NCC classification accuracy rates for every intermediate layer of a network of a specific depth. We made several interesting observations: **(i)** Networks with eight or more hidden layers display NCC train accuracy rates of about 100% in the eighth and higher layers, indicating that they are effectively of depth 7. **(ii)** The top layer embeddings become NCC separable at approximately the same epoch. **(iii)** The degree of NCC separability of intermediate layer i converges as a function of L. In other words, the degree of NCC separability for each layer is more or less the same across all neural networks with a depth of at least 8, regardless of whether the layer is at the beginning or the end of the network.

For additional experiments and repeat results with various architectures and datasets, refer to Figs. 4-13 in Appendix A. In these experiments, we also report NCC train and test accuracy rates, along with additional measures of neural collapse when varying the depth. For instance, in Figs. 4 and 5, we present the outcomes with CONVRES-L-500.

The effect of the depth on the ϵ-effective depth. In Obs. 1 we claimed that the ϵ-effective depth is insensitive to the actual depth of the network (once it exceeds a certain threshold). To validate this hypothesis we conducted the following experiments. We trained models on MNIST, Fashion MNIST and CIFAR10 with varying depth L. In Fig. 2 we plotted the averaged ϵ-effective depths of each network's last $k = 1, 20$ epochs as a function of ϵ. We also average the results across 5 different weight initializations and plot them along with error bar standard deviations. As can be seen, the ϵ-effective depth is almost unaffected by the choice of L for a given ϵ. Remarkably, for each ϵ, the averaged effective depth varies very little across the various networks. Differently said, the ϵ-effective depths of two trained deep networks of different depths are more or less the same, validating our Minimal Depth Hypothesis.

NCC separability with partially corrupted labels. Simply put, Prop. 1 compares the depths required to fit correct labels and partially corrupt labels. To better understand the effect of corrupted labels on the complexity of the task, we compare the ϵ-effective depths of models trained with varying amounts of corrupted labels. Namely, we study the *degree* of NCC separability in the intermediate layers of neural networks that are trained with varying amounts of corrupted labels.

For this experiment, we trained instances of CONV-10-400 for CIFAR10 classification with 0%, 10% and 75% corrupted labels (e.g., uniformly distributed random labels). We plot the degrees of NCC separation on the train and test sets, $1 - \text{err}_S(\hat{h}_i)$ and $1 - \text{err}_P(\hat{h}_i)$, across the intermediate layers of the neural networks during the optimization procedure.

As can be seen in Fig. 3, when increasing the number of random labels, the degree of NCC separability across the intermediate layers tends to decrease. For example, when training with $\geq 25\%$ corrupted labels, the sixth layer's NCC accuracy rate drops lower than 98%, in comparison with training without corrupted labels that gives us $> 98\%$ accuracy. In particular, the ϵ-effective depth of the former network is 6 while the latter's is 5 when $\epsilon = 0.02$ (see Def. 1). This experiment is extended and repeated in a variety of settings in Figs. 14-18.

Estimating the bound in equation 4. We estimate the bound in equation 4 for multiple architectures and datasets. In each case we used $\epsilon = 0.005$ by default and employed different 'guesses' p (see Tab. 2) depending on the complexity of the learning task. We report an estimation of the expected test error of the models, $\mathbb{E}_{S_1,\gamma}[\text{err}_P(h^{\gamma}_{S_1})]$ and an estimation of the bound for each selection of p. For concrete technical details, see Appendix A.

As can be seen, for appropriate choices of p, we obtained non-trivial estimates of the test performance of the models, which is uncommon for standard bounds for deep neural networks. As expected, if the value of p is too optimistic (e.g., close to $\mathbb{E}_{S_1,\gamma}[\text{err}_P(h^{\gamma}_{S_1})]$), then, the first term in the bound tends to be large compared to $\mathbb{E}_{S_1,\gamma}[\text{err}_P(h^{\gamma}_{S_1})]$. As predicted, when p is increased, the first term in the bound tends to decrease.

Comparing our bound with standard generalization bounds. We expect the bound in equation 4 to be insensitive to depth because the ϵ-effective depth of deep neural networks is insensitive to depth, as shown in Fig. 2. We estimated the bound for various models and datasets, including CONV-L-50 trained on MNIST and CONV-L-100 trained on Fashion MNIST and CIFAR10, and CONVRES-L-50 trained on CIFAR10 with different values of L. The results, shown in Tab. 1, indicate that our bound gives similar values for each value of L. We also compared our bound to several norm-based generalization bounds for deep networks that can be found in (Bartlett & Mendelson, 2003; Neyshabur et al., 2015; Bartlett et al., 2017; Neyshabur et al., 2019) (we used the implementation of Neyshabur et al. (2019) to compute them). We found that our bound outperforms traditional bounds, as it is empirically non-vacuous and fairly independent of depth, while traditional bounds are extremely vacuous and rapidly increase with depth. These results support our prediction of the superiority of our bound over traditional bounds[1].

[1] The norm-based generalization bounds could not be calculated for the CONVRES-L-50 architecture, as these bounds are not applicable for neural networks incorporating residual connections.

5 Conclusions

Understanding the ability of SGD to generalize well when training overparameterized neural network is attributed as one of the major open problems in deep learning theory (Zhang et al., 2017). In this paper we offer a new angle to study the role of depth in deep learning and the connection between neural collapse and generalization.

Our approach involves introducing the concept of effective depth, which identifies the lowest layer that exhibits NCC separability. We propose a novel generalization bound that estimates the likelihood that the effective depth of a trained neural network is strictly smaller than the minimal depth required to achieve NCC separability with partially corrupted labels. As demonstrated empirically, this criterion is a useful predictor of generalization. Furthermore, we characterize and empirically demonstrate that when sufficiently deep networks are trained, they converge to the same effective depth, implying that our bound is fairly constant when the depth is varied.

References

Madhu S. Advani and Andrew M. Saxe. High-dimensional dynamics of generalization error in neural networks, 2017. URL `https://arxiv.org/abs/1710.03667`.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *ArXiv*, abs/1610.01644, 2017.

Kai Arulkumaran, Antoine Cully, and Julian Togelius. Alphastar: An evolutionary computation perspective, 2019. URL `http://arxiv.org/abs/1902.01724`. cite arxiv:1902.01724.

Peter L. Bartlett and Philip M. Long. Failures of model-dependent generalization bounds for least-norm interpolation. *Journal of Machine Learning Research*, 22(204):1–15, 2021. URL `http://jmlr.org/papers/v22/20-1164.html`.

Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *J. Mach. Learn. Res.*, 3(null):463–482, mar 2003. ISSN 1532-4435.

Peter L. Bartlett, Dylan J. Foster, and Matus Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6241–6250, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Peter L. Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, 20 (63):1–17, 2019. URL `http://jmlr.org/papers/v20/17-612.html`.

Mikhail Belkin. Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation. *Acta Numerica*, 30:203 – 248, 2021.

Mikhail Belkin, Siyuan Ma, and Soumik Mandal. To understand deep learning we need to understand kernel learning. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 541–549. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/belkin18a.html`.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32): 15849–15854, 2019. doi: 10.1073/pnas.1903070116. URL `https://www.pnas.org/doi/abs/10.1073/pnas.1903070116`.

Ido Ben-Shaul and Shai Dekel. Sparsity-probe: Analysis tool for deep learning models. *ArXiv*, abs/2105.06849, 2021.

Ido Ben-Shaul and Shai Dekel. Nearest class-center simplification through intermediate layers. *PMLR*, 196: 37–47, 2022. URL `https://proceedings.mlr.press/v196/ben-shaul22a.html`.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir

Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

Gilad Cohen, Guillermo Sapiro, and Raja Giryes. Dnn or k-nn: That is the generalize vs. memorize question, 2018. URL https://arxiv.org/abs/1805.06822.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, jun 2019.

Tolga Ergen and Mert Pilanci. Revealing the structure of deep neural networks via convex duality. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 3004–3014. PMLR, 18–24 Jul 2021.

Cong Fang, Hangfeng He, Qi Long, and Weijie J. Su. Exploring deep neural networks via layer-peeled model: Minority collapse in imbalanced training. *Proceedings of the National Academy of Sciences*, 118(43), 2021.

Tomer Galanti, András György, and Marcus Hutter. On the role of neural collapse in transfer learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=SwIp410B6aQ.

Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-independent sample complexity of neural networks. *Information and Inference: A Journal of the IMA*, 9, 12 2017. doi: 10.1093/imaiai/iaz007.

Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJgIPJBFvH.

Jianfeng Lu and Stefan Steinerberger. Neural collapse with cross-entropy loss. *CoRR*, abs/2012.08465, 2020.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012. ISBN 026201825X.

Grégoire Montavon, Mikio L. Braun, and Klaus-Robert Müller. Kernel analysis of deep networks. *J. Mach. Learn. Res.*, 12:2563–2581, 2011.

Vaishnavh Nagarajan and J. Zico Kolter. Uniform convergence may be unable to explain generalization in deep learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/05e97c207235d63ceb1db43c60db7bbb-Paper.pdf.

Jeffrey Negrea, Gintare Karolina Dziugaite, and Daniel M. Roy. In defense of uniform convergence: Generalization via derandomization with an application to interpolating predictors. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In Peter Grünwald, Elad Hazan, and Satyen Kale (eds.), *Proceedings of The 28th Conference on Learning Theory*, volume 40 of *Proceedings of Machine Learning Research*, pp. 1376–1401, Paris, France, 03–06 Jul 2015. PMLR. URL https://proceedings.mlr.press/v40/Neyshabur15.html.

Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A PAC-bayesian approach to spectrally-normalized margin bounds for neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Skz_WfbCZ.

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. *ArXiv*, abs/1805.12076, 2019.

Vardan Papyan. Traces of class/cross-class structure pervade deep learning spectra. *Journal of Machine Learning Research*, 21(252):1–64, 2020. URL `http://jmlr.org/papers/v21/20-933.html`.

Vardan Papyan, Yaniv Romano, and Michael Elad. Convolutional neural networks analyzed via convolutional sparse coding. *J. Mach. Learn. Res.*, 18:83:1–83:52, 2017.

Vardan Papyan, X. Y. Han, and David L. Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020.

Akshay Rangamani, Mengjia Xu, Andrzej Banburski, Qianli Liao, Tomer Galanti, and Tomaso Poggio. Dynamics and neural collapse in deep classifiers trained with the square loss. Technical report, Center for Brains, Minds and Machines (CBMM), 2022.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning - From Theory to Algorithms.* Cambridge University Press, 2014. ISBN 978-1-10-705713-5.

Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *ArXiv*, abs/1703.00810, 2017.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–489, 2016. ISSN 0028-0836. doi: 10.1038/nature16961.

Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

Tom Tirer and Joan Bruna. Extended unconstrained features model for exploring deep neural collapse, 2022. URL `https://arxiv.org/abs/2202.08087`.

Vladimir N. Vapnik. *Statistical Learning Theory.* Wiley-Interscience, 1998.

Zitong Yang, Yu Bai, and Song Mei. Exact gap between generalization error and uniform convergence in random feature models. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11704–11715. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/yang21a.html`.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers, 2021.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=Sy8gdB9xx`.

Lijia Zhou, Danica J. Sutherland, and Nati Srebro. On uniform convergence and low-norm interpolation learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6867–6877. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/4cc5400e63624c44fadeda99f57588a6-Paper.pdf`.

Zhihui Zhu, Tianyu DING, Jinxin Zhou, Xiao Li, Chong You, Jeremias Sulam, and Qing Qu. A geometric analysis of neural collapse with unconstrained features. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=KRODJAa6pzE`.

A Additional Experiments and Details

A.1 Architectures

In this section, we describe the architectures used in our experiments.

The first architecture is a convolutional network, denoted CONV-L-H, which consists of a stack of two 2×2 convolutional layers with stride 2, batch normalization, and ReLU activation. This is followed by L stacks of blocks $g^i(x) = \sigma(B_i(C_i(x)))$, where C_i is a 3×3 convolutional layer with H channels, stride 1, and padding 1, B_i is a batch normalization layer, and σ is the ReLU activation. The final layer is linear. The ith intermediate layer refers to the output of the ith block of g^i.

The second architecture is an MLP, denoted MLP-L-H, which consists of L hidden layers, where each layer $g^i(x) = \sigma(B_i(T_i(x)))$ contains a linear layer T_i with output width H, followed by a batch normalization layer B_i and a ReLU activation function σ. The final layer is linear.

The third architecture is a convolutional residual network, denoted CONVRES-L-H. It consists of a stack of two 2×2 convolutional layers with stride 2, batch normalization, and ReLU activation, followed by L residual blocks. Each block computes $g^i(x) = \sigma(x + B^2_i(C^2_i(\sigma(B^1_i(C^1_i(x))))))$, where C^j_i is a 3×3 convolutional layer with H channels, stride 1, and padding 1, B^j_i is a batch normalization layer, and σ is the ReLU activation. The final layer is linear.

A.2 Estimating the Generalization Bound

In this section we describe how we empirically estimate the bound in Prop. 1.

Estimating the bound. We would like to estimate the first term in the bound,

$$\mathbb{P}_{S_1,S_2,\tilde{Y}_2}\left[\mathbb{E}_\gamma[\mathscr{d}^\epsilon_{S_1}(h^\gamma_{S_1})] \geq \mathscr{d}^\epsilon_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2)\right]. \tag{5}$$

According to Prop. 2 in order to estimate this term we need to generate i.i.d. triplets $(S^i_1, S^i_2, \tilde{Y}^i_2)$. Since we have limited access to training data, we use a variation of cross-validation and generate $k_1 = 5$ i.i.d. disjoint splits (S^i_1, S^i_2) of the training data S. For each one of these pairs, we generate $k_2 = 3$ corrupted labelings $\tilde{Y}^{ij}_2$. We denote by $\tilde{S}^{ij}_2$ the set obtained by replacing the labels of S^i_2 with $\tilde{Y}^{ij}_2$ and $\tilde{S}^{ij}_3 := S^i_1 \cup \tilde{S}^{ij}_2$.

As a first step, we would like to estimate $\mathbb{E}_\gamma[\mathscr{d}^\epsilon_{S^i_1}(h^\gamma_{S^i_1})]$ for each $i \in [k_1]$. For this purpose, we randomly select $T_1 = 5$ different initializations $\gamma_1, \dots, \gamma_{T_1}$ and for each one, we train the model $h^{\gamma_t}_{S^i_1}$ using the training protocol described in Sec. 4.1. Once trained, we compute $\mathscr{d}^\epsilon_{S_1}(h^{\gamma_t}_{S^i_1})$ for each $t \in [T_1]$ (see Def. 1) and approximate $\mathbb{E}_\gamma[\mathscr{d}^\epsilon_{S^i_1}(h^\gamma_{S^i_1})]$ using $d_i := \frac{1}{T_1}\sum_{t=1}^{T_1} \mathscr{d}^\epsilon_{S^i_1}(h^{\gamma_t}_{S^i_1})$.

As a next step, we would like to evaluate $\mathbb{I}[d_i \geq \mathscr{d}^\epsilon_{\min}(\mathcal{G}, \tilde{S}^{ij}_3)]$. We notice that $d_i \geq \mathscr{d}^\epsilon_{\min}(\mathcal{G}, S^i_1 \cup \tilde{S}^i_2)$ if and only if there is a d_i-layered neural network $f = g^{d_i} \circ \dots \circ g^1$ for which $\mathrm{err}_{\tilde{S}^{ij}_3}(\hat{h}) \leq \epsilon$, where $\hat{h}(x) := \arg\min_{c \in [C]} \|f(x) - \mu_f(S_c)\|$. In general, computing this Boolean value is computationally hard. Therefore, to estimate this Boolean value, we simply train a $(d_i + 1)$-layered network $h = e \circ f$ and check whether its penultimate layer is ϵ-NCC separable, i.e., $\mathrm{err}_{\tilde{S}^{ij}_3}(\hat{h}) \leq \epsilon$, where $\hat{h}(x) := \arg\min_{c \in [C]} \|f(x) - \mu_f(S_c)\|$. If SGD implicitly optimizes neural networks to maximize NCC separability as observed in (Papyan et al., 2020) (and also in this paper), we should expect to obtain ϵ-NCC separability in the penultimate layer if that is possible with a d_i-layered network. Since training might be non-optimal, to obtain a robust estimation, we train $T_2 = 5$ models $h_t = e_t \circ f_t$ of depth $d_i + 1$ and pick the one with the best NCC separability in its penultimate layer. Namely, we replace $\mathscr{d}^\epsilon_{\min}(\mathcal{G}, \tilde{S}^{ij}_3)$ with $\min_{t \in [T_2]} \mathscr{d}^\epsilon_{\tilde{S}^{ij}_3}(h_t)$ and estimate $\mathbb{I}[d_i \geq \mathscr{d}^\epsilon_{\min}(\mathcal{G}, \tilde{S}^{ij}_3)]$ using $\mathbb{I}[d_i \geq \min_{t \in [T_2]} \mathscr{d}^\epsilon_{\tilde{S}^{ij}_3}(h_t)]$.

Our final estimation is the following

$$\frac{1}{k_1}\sum_{i=1}^{k_1}\frac{1}{k_2}\sum_{j=1}^{k_2}\mathbb{I}\left[d_i \geq \min_{t \in [T_2]} \mathscr{d}^\epsilon_{\tilde{S}^{ij}_3}(h_t)\right] \approx \mathbb{P}_{S_1,S_2,\tilde{Y}_2}\left[\mathbb{E}_\gamma[\mathscr{d}^\epsilon_{S_1}(h^\gamma_{S_1})] \geq \mathscr{d}^\epsilon_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2)\right]. \tag{6}$$

In order to estimate the bound we assume that δ^1_m and $\delta^2_{m,p,\alpha}$ are negligible constants and that $\alpha = 1$. The estimation of the bound is given by the sum of the left hand side in equation 6 and p.

Estimating the mean test error. To estimate the mean test error, $\mathbb{E}_{S_1,\gamma}[\mathrm{err}_P(h^{\gamma}_{S_1})]$, as typically done in machine learning, we replace the population distribution P with the test set S_{test} and we replace the expectation over S_1 and γ with averages across the $k_1 = 5$ random selections of $\{S^i_1\}^{k_1}_{i=1}$ and $T_1 = 5$ random selections of $\{\gamma_t\}^{T_1}_{t=1}$. Namely, we compute the following $\frac{1}{k_1}\sum^{k_1}_{i=1}\frac{1}{T_1}\sum^{T_1}_{t=1}\mathrm{err}_{S_{test}}(h^{\gamma_t}_{S^i_1}) \approx \mathbb{E}_{S_1,\gamma}[\mathrm{err}_P(h^{\gamma}_{S_1})]$.

A.3 Neural Collapse

To obtain a comprehensive analysis of collapse across layers, we also estimate the degree of NC1.

To evaluate NC1, we follow the process suggested by Galanti et al. (2022), which is a simplified version of the original approach of Papyan et al. (2020). For a feature map $f : \mathbb{R}^d \to \mathbb{R}^p$ and two (class-conditional) distributions[2] Q_1, Q_2 over $\mathcal{X} \subset \mathbb{R}^d$, we define their *class-distance normalized variance* (CDNV) to be

$$V_f(Q_1, Q_2) \ := \ \frac{\mathrm{Var}_f(Q_1) + \mathrm{Var}_f(Q_2)}{2\|\mu_f(Q_1) - \mu_f(Q_2)\|^2},$$

where $\mu_u(Q) := \mathbb{E}_{x\sim Q}[u(x)]$ and by $\mathrm{Var}_u(Q) := \mathbb{E}_{x\sim Q}[\|u(x) - \mu_u(Q)\|^2]$ the mean and variance of $u(x)$ for $x \sim Q$. Essentially, this quantity measures to what extent the feature vectors of samples from Q_1 and Q_2 are separated and clustered in space.

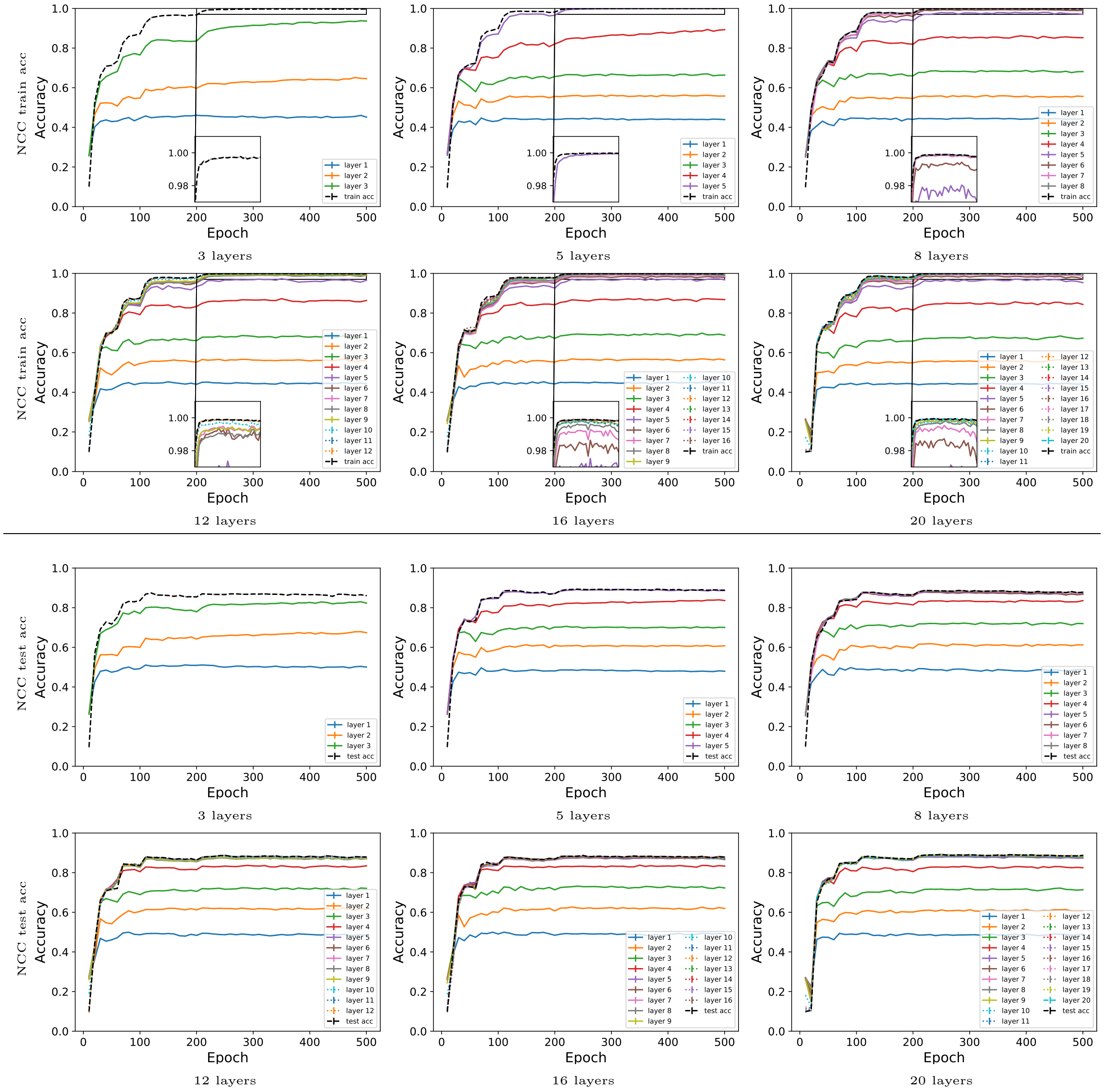
To demonstrate the gradual evolution of collapse across the layers, for each sub-architecture $f^i = g^i \circ \cdots \circ g^1(x)$ we consider the train and test class features variations $\mathrm{Avg}_{c\neq c'}[V_{f^i}(S_c, S_{c'})]$ and $\mathrm{Avg}_{c\neq c'}[V_{f^i}(P_c, P_{c'})]$. The population distribution of each class, P_c, is replaced with the test samples of that class.

As shown by Galanti et al. (2022), this definition is essentially the same as that of Papyan et al. (2020). Furthermore, they showed that the NCC classification error rate can be upper bounded in terms of the CDNV. However, the NCC error can be zero in cases where the CDNV is larger than zero. For example, if the two classes are uniformly distributed over the 1-radius circles around the points $(-1, 0)$ and $(1, 0)$ in $\mathbb{R}^2$, then they are perfectly NCC separable while the CDNV between the two distributions is 0.25.

Auxiliary experiments on the effective depth. In Figs. 6-13 we plot the NCC and the CDNV rates of neural networks with varying numbers of layers evaluated on the train and test data. Each curve stands for a different layer within the network. As can be seen, in all cases, for networks deeper than a threshold we obtain (near-perfect) NCC separability in all of the top layers. Furthermore, the degree of class-features variability collapse increases with the network's depth as depicted by decreasing CDNVs.

Auxiliary experiments with noisy labels. In Figs. 14-18 we repeat the experiment in Fig. 3 and plot the results of the same experiment, with different networks and datasets (see captions). As can be seen, the effective NCC depth of a neural network tends to increase as we train with increasing amounts of corrupted labels.

[2] The definition can be extended to finite sets $S_1, S_2 \subset \mathcal{X}$ by defining $V_f(S_1, S_2) = V_f(U[S_1], U[S_2])$.



3 layers

5 layers

8 layers

12 layers

16 layers

20 layers

3 layers

5 layers

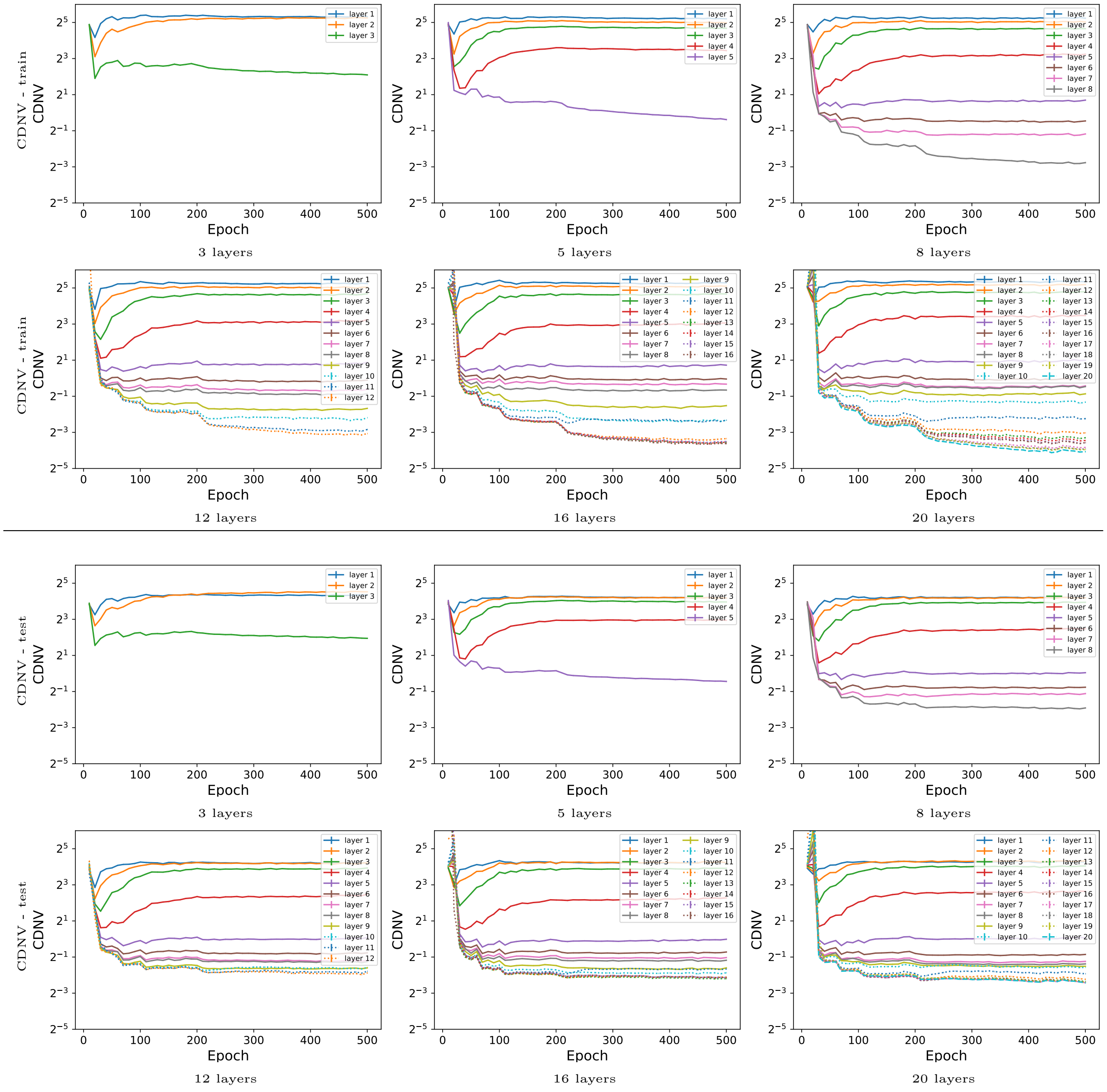
8 layers

12 layers

16 layers

20 layers

Figure 4: **Intermediate NCC separability of CONVRES-L-500 trained on CIFAR10.** We plot the NCC train and test accuracy rates of neural networks with varying numbers of layers. Each curve stands for a different layer within the network.



3 layers

5 layers

8 layers

12 layers

16 layers

20 layers

3 layers

5 layers

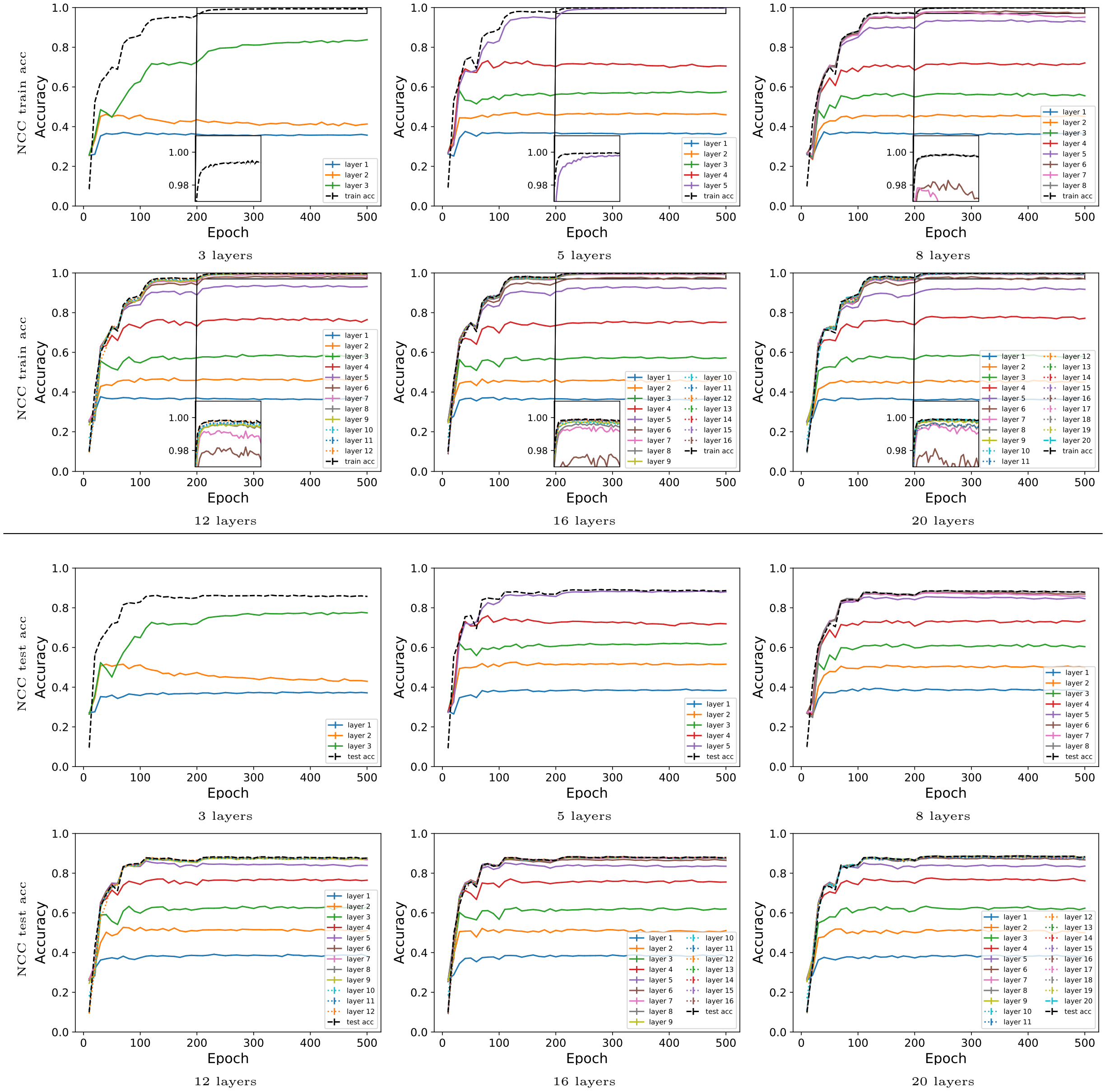
8 layers

12 layers

16 layers

20 layers

Figure 5: **Intermediate class-features variability collapse separability of CONVRES-*L*-500 trained on CIFAR10.** We plot the CDNV on the training and test data of neural networks with varying numbers of layers. Each curve stands for a different layer within the network.



3 layers

5 layers

8 layers

12 layers

16 layers

20 layers

3 layers

5 layers

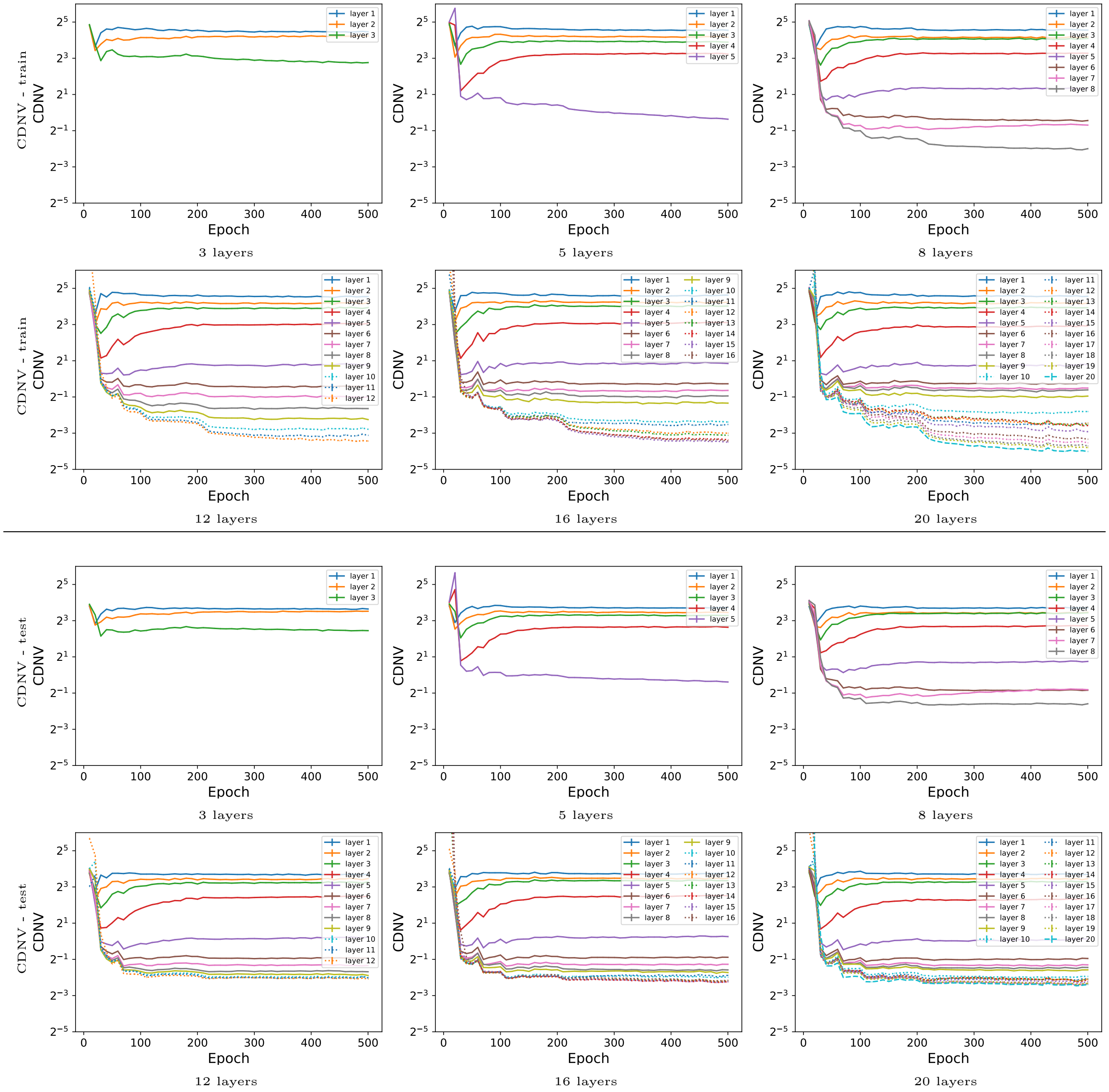
8 layers

12 layers

16 layers

20 layers

Figure 6: **Intermediate NCC separability of CONV-*L*-400 trained on CIFAR10.** We plot the NCC train and test accuracy rates of neural networks with varying numbers of layers. Each curve stands for a different layer within the network.



3 layers

5 layers

8 layers

12 layers

16 layers

20 layers

3 layers

5 layers

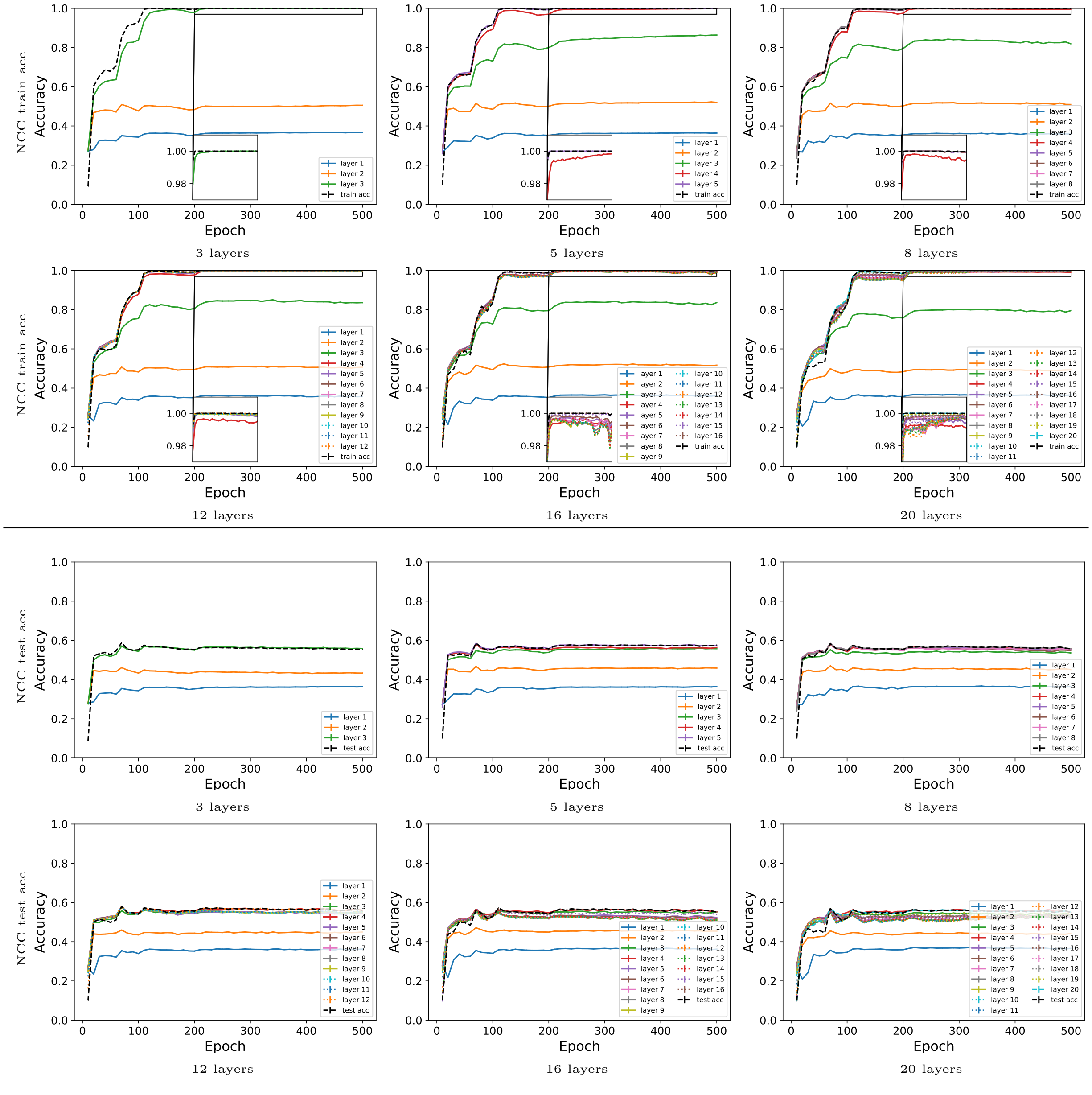
8 layers

12 layers

16 layers

20 layers

Figure 7: **Intermediate class-features variability collapse separability of CONV-L-400 trained on CIFAR10.** We plot the CDNV on the training and test data of neural networks with varying numbers of layers. Each curve stands for a different layer within the network.



3 layers

5 layers

8 layers

12 layers

16 layers

20 layers

3 layers

5 layers

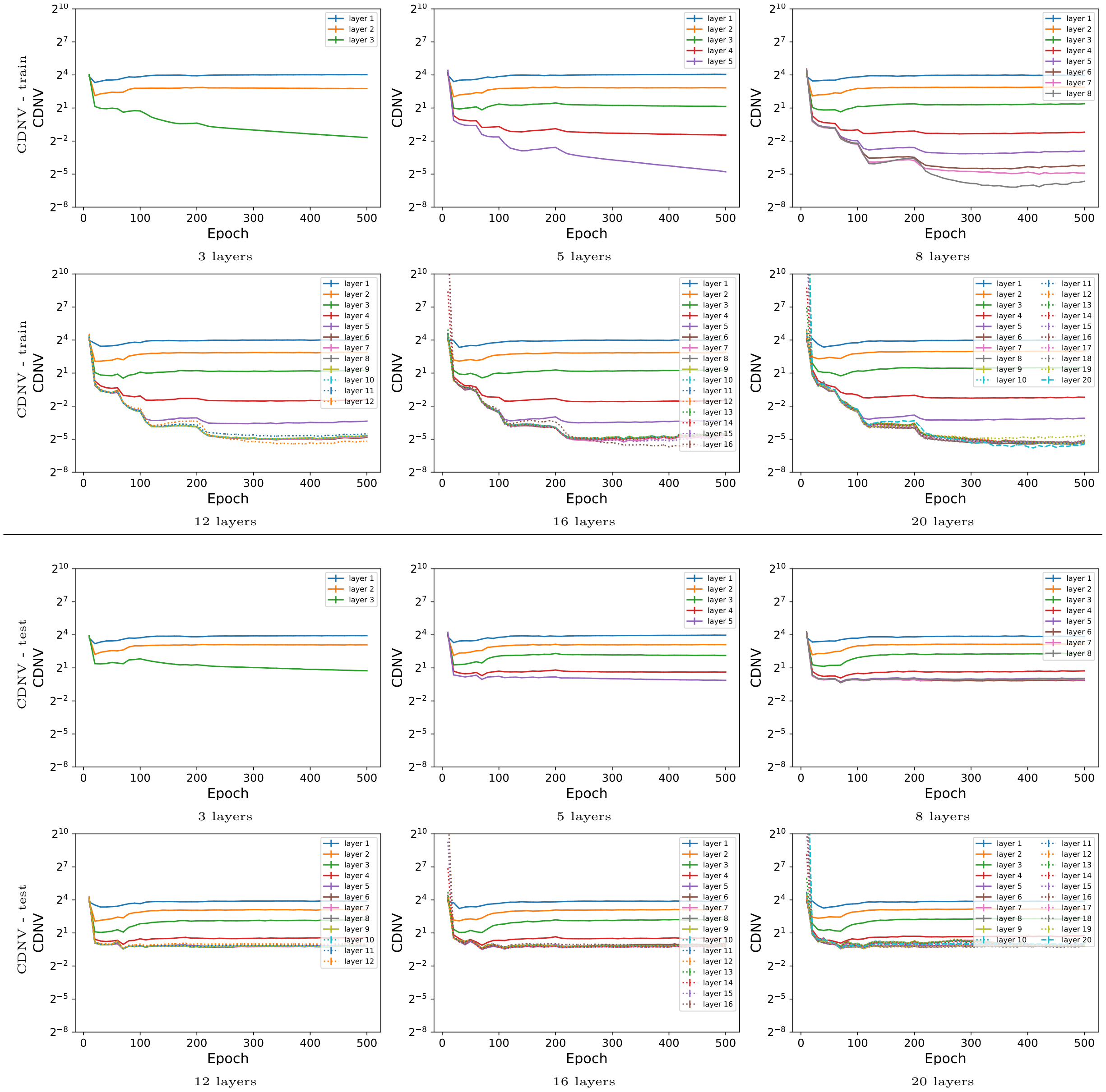
8 layers

12 layers

16 layers

20 layers

Figure 8: **Intermediate neural collapse of MLP-L-300 trained on CIFAR10.** We plot the NCC train and test accuracy rates of neural networks with varying numbers of layers. Each curve stands for a different layer within the network.



3 layers

5 layers

8 layers

12 layers

16 layers

20 layers

3 layers

5 layers

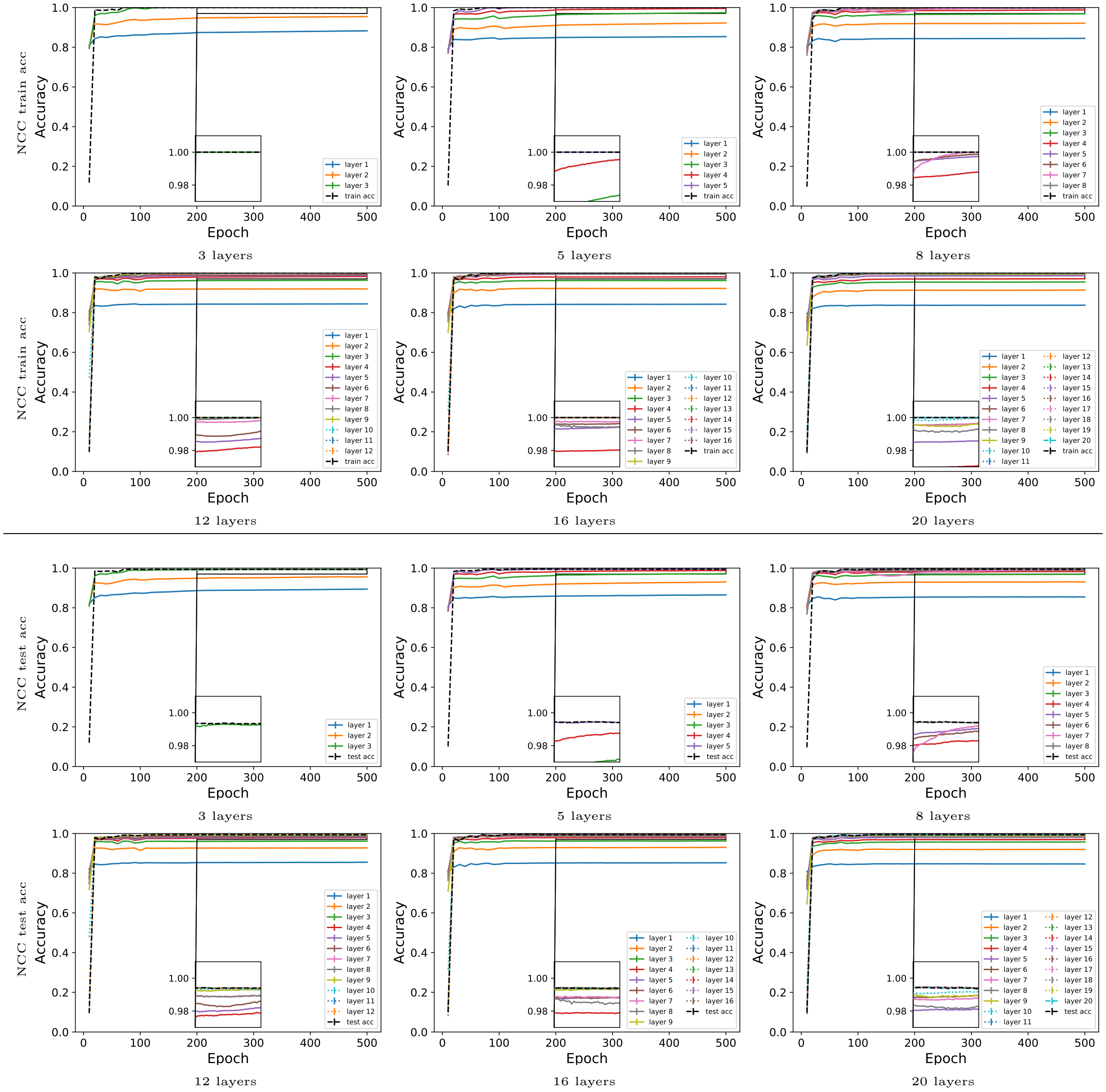
8 layers

12 layers

16 layers

20 layers

Figure 9: **Intermediate class-features variability collapse separability of MLP-L-300 trained on CIFAR10.** We plot the CDNV on the training and test data of neural networks with varying numbers of layers. Each curve stands for a different layer within the network.



3 layers

5 layers

8 layers

12 layers

16 layers

20 layers

3 layers

5 layers

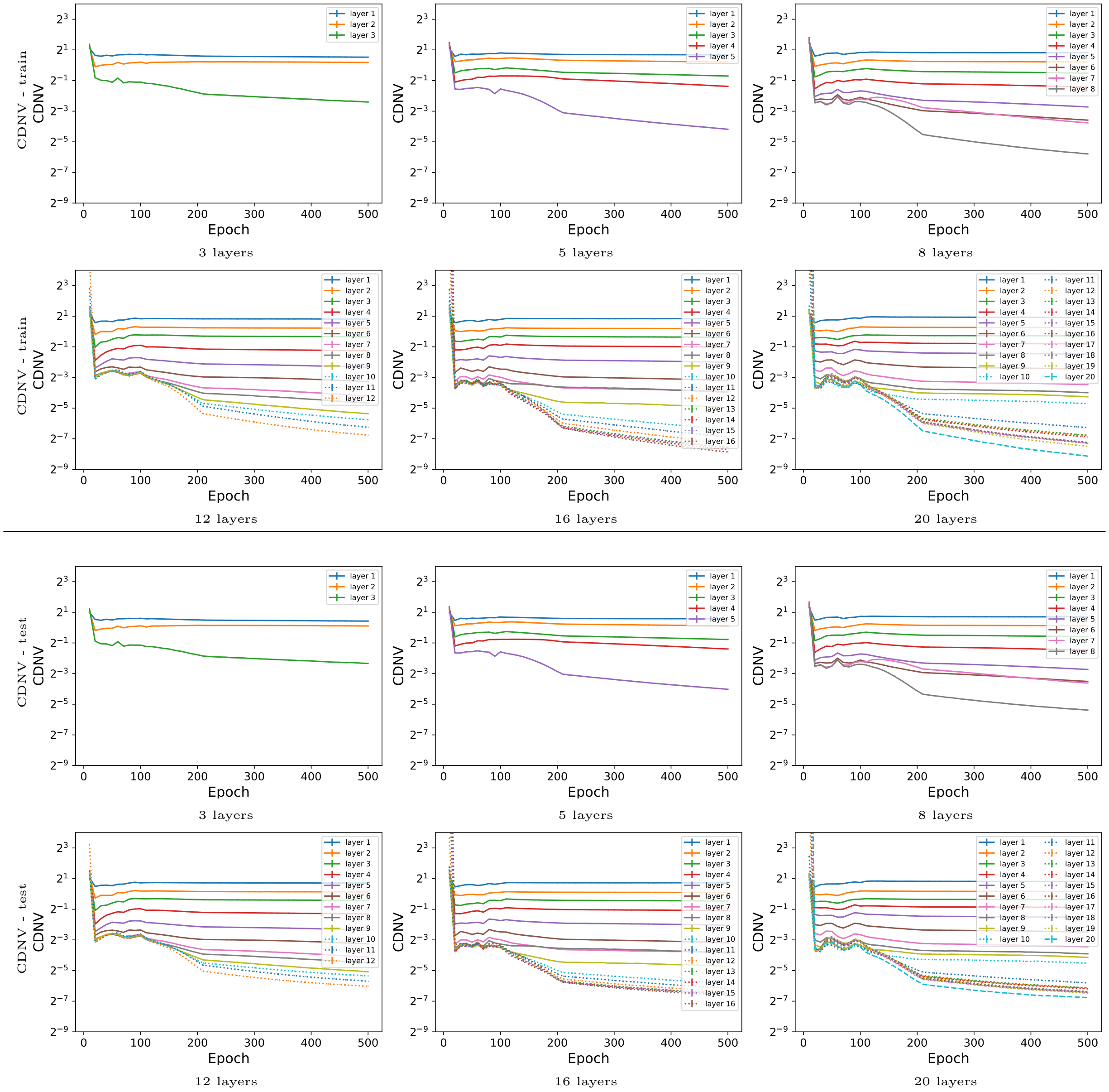
8 layers

12 layers

16 layers

20 layers

Figure 10: **Intermediate neural collapse of CONV-*L*-50 trained on MNIST.** We plot the NCC train and test accuracy rates of neural networks with varying numbers of layers. Each curve stands for a different layer within the network.



3 layers

5 layers

8 layers

12 layers

16 layers

20 layers

3 layers

5 layers

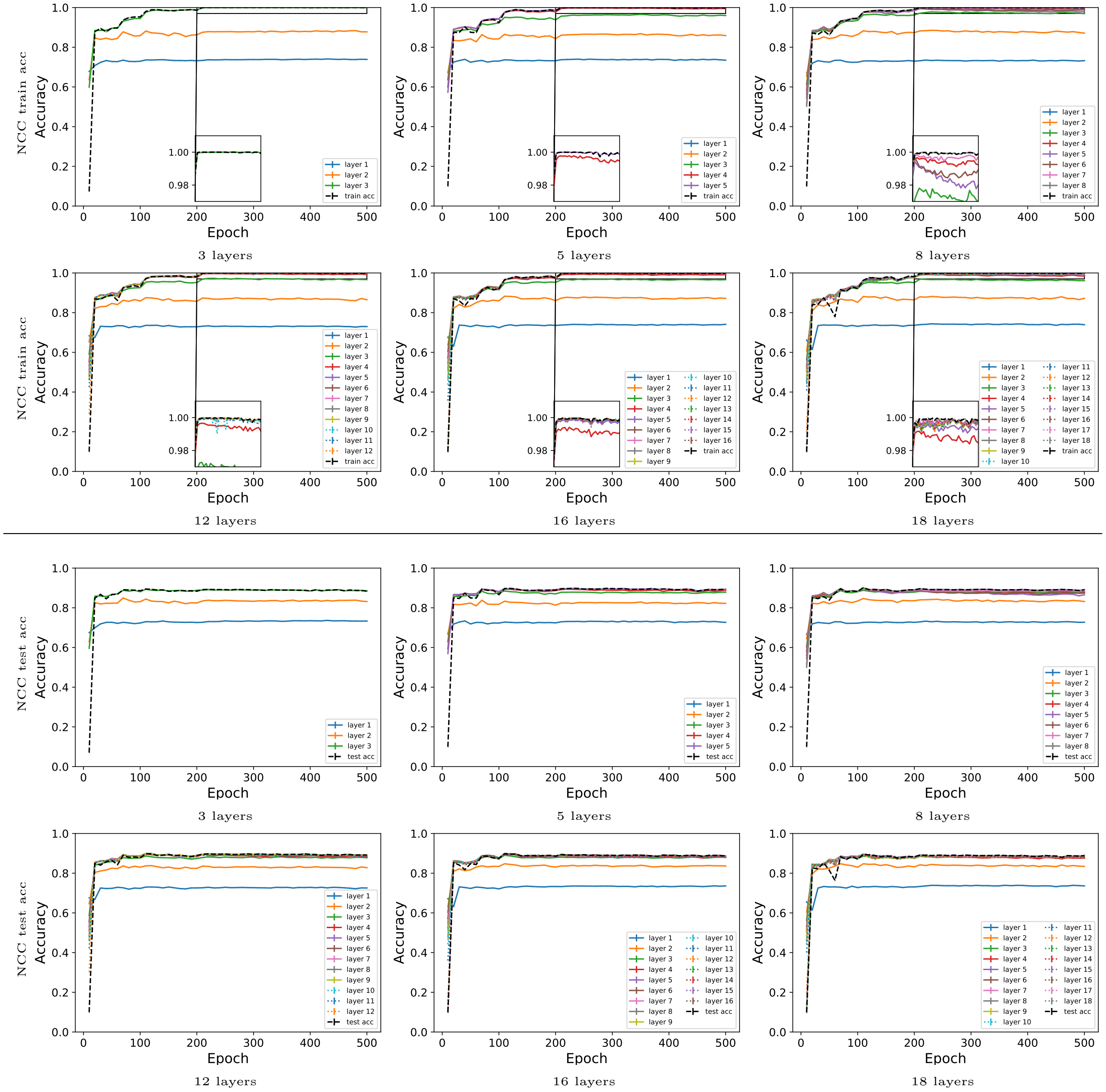
8 layers

12 layers

16 layers

20 layers

Figure 11: **Intermediate class-features variability collapse separability of CONV-L-50 trained on MNIST.** We plot the CDNV on the training and test data of neural networks with varying numbers of layers. Each curve stands for a different layer within the network.



3 layers

5 layers

8 layers

12 layers

16 layers

18 layers

3 layers

5 layers

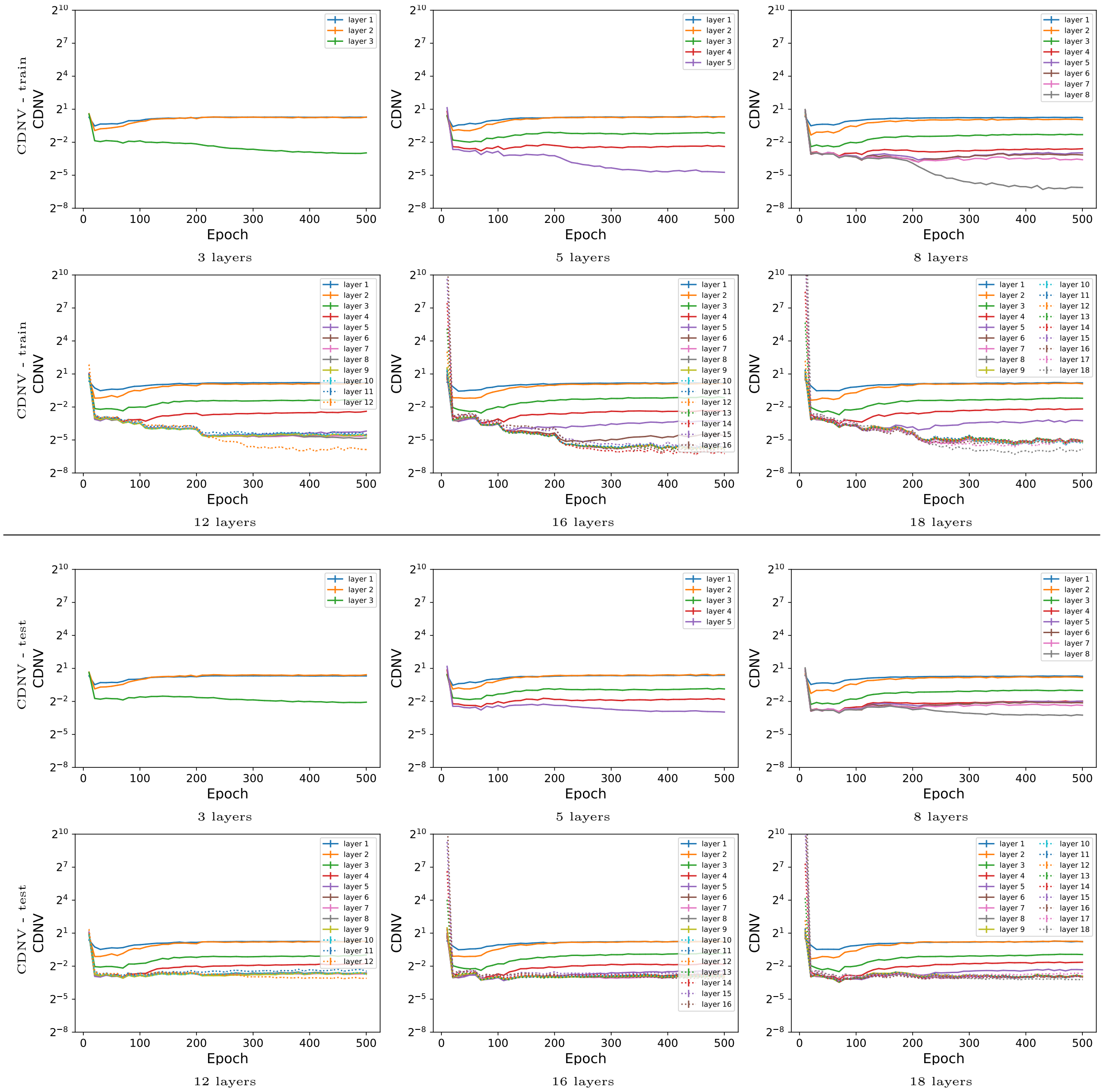
8 layers

12 layers

16 layers

18 layers

Figure 12: **Intermediate neural collapse of MLP-L-100 trained on Fashion MNIST.** We plot the NCC train and test accuracy rates of neural networks with varying numbers of layers. Each curve stands for a different layer within the network.



3 layers

5 layers

8 layers

12 layers

16 layers

18 layers

3 layers

5 layers

8 layers

12 layers

16 layers

18 layers

Figure 13: **Intermediate class-features variability collapse separability of MLP-L-100 trained on Fashion MNIST.** We plot the CDNV on the training and test data of neural networks with varying numbers of layers. Each curve stands for a different layer within the network.

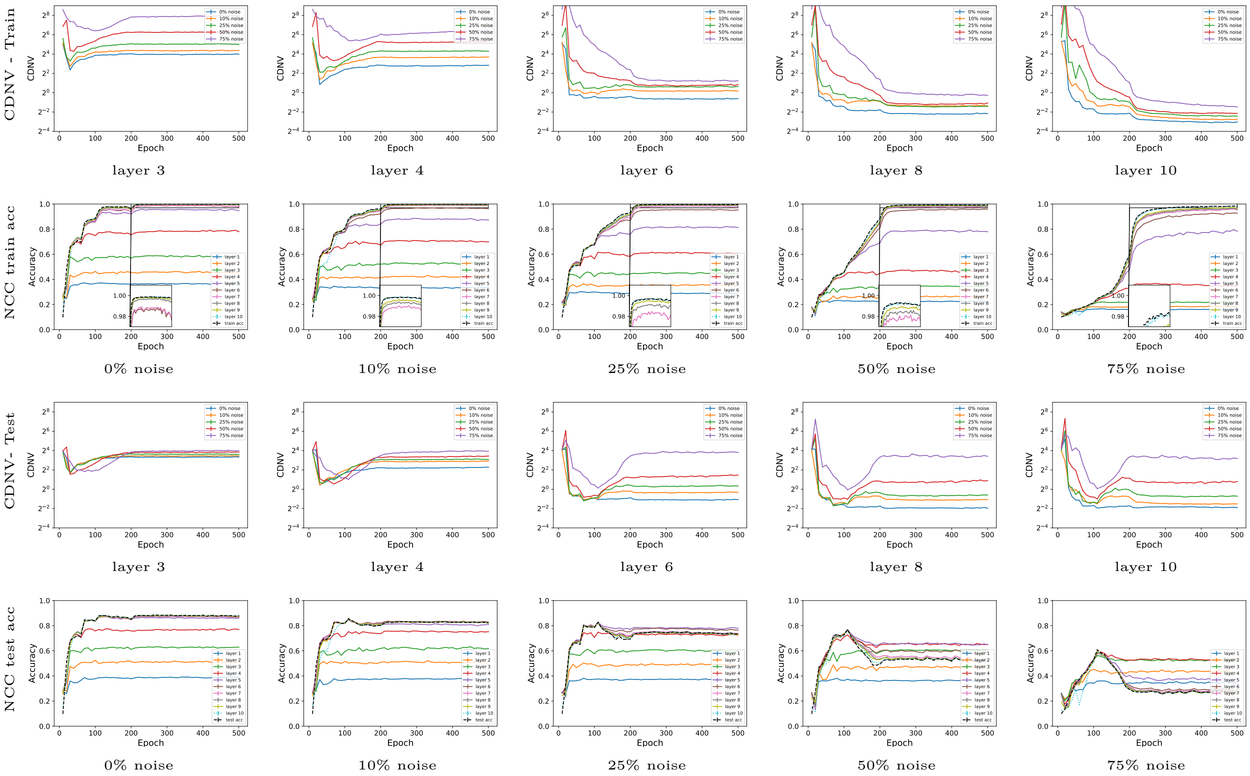
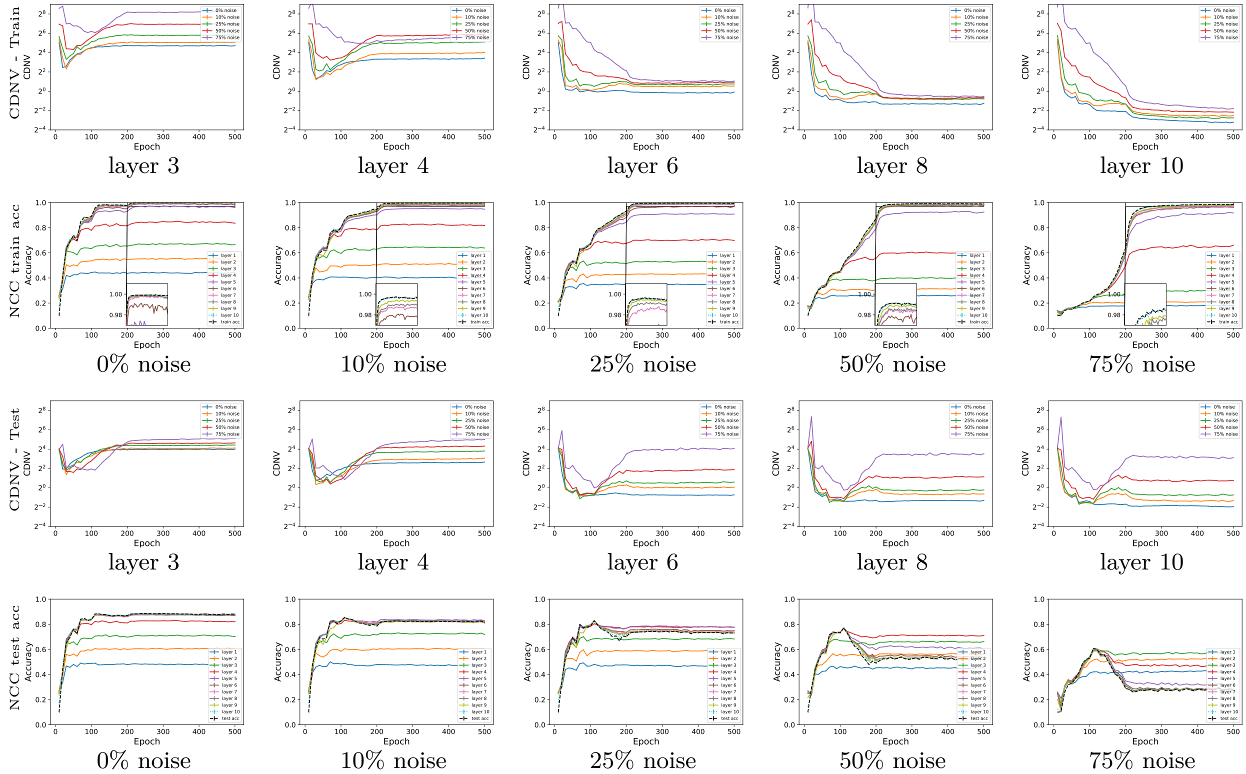


Figure 14: **Intermediate neural collapse of CONV-10-400 trained on CIFAR10 with partially corrupted labels.** In the first (third) row, we plot the CDNV on the train (test) data for intermediate layers of networks trained with varying amounts of corrupted labels (see legend). In the second (fourth) row, we plot the NCC accuracy rates of the various layers of a network trained with a certain amount of corrupted labels (see titles).

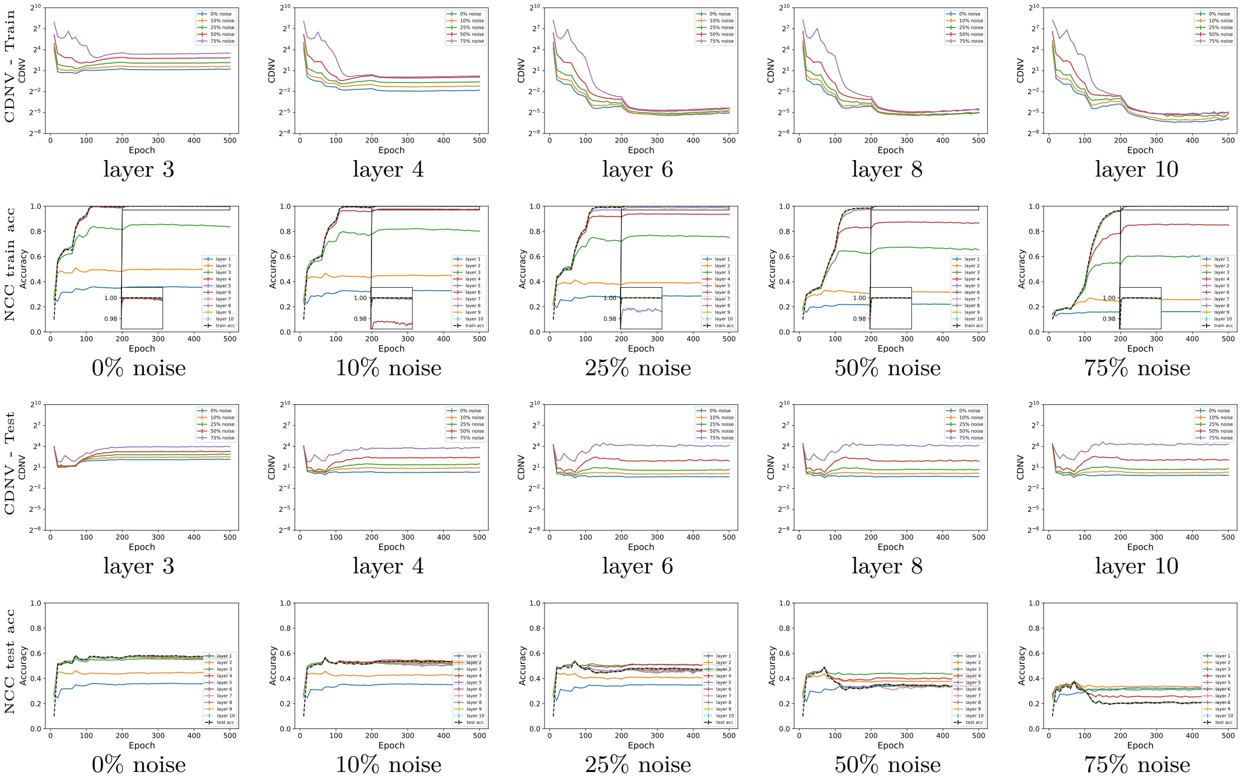

layer 3 layer 4 layer 6 layer 8 layer 10

0% noise 10% noise 25% noise 50% noise 75% noise

layer 3 layer 4 layer 6 layer 8 layer 10

0% noise 10% noise 25% noise 50% noise 75% noise

Figure 15: **Intermediate neural collapse of CONVRES-10-500 trained on CIFAR10 with noisy labels.** See Fig 3 in the main text for details.



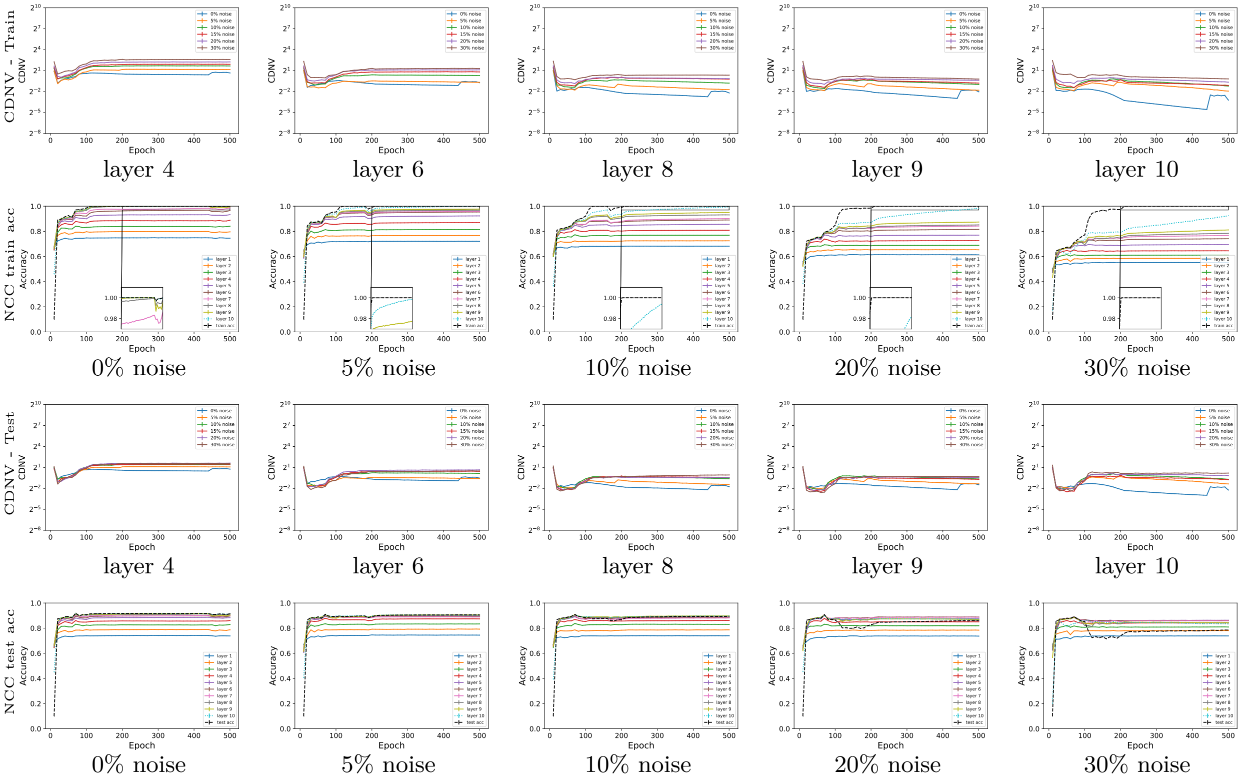
layer 3 layer 4 layer 6 layer 8 layer 10

0% noise 10% noise 25% noise 50% noise 75% noise

layer 3 layer 4 layer 6 layer 8 layer 10

0% noise 10% noise 25% noise 50% noise 75% noise

Figure 16: **Intermediate neural collapse of MLP-10-500 trained on CIFAR10 with noisy labels.** See Fig 3 in the main text for details.



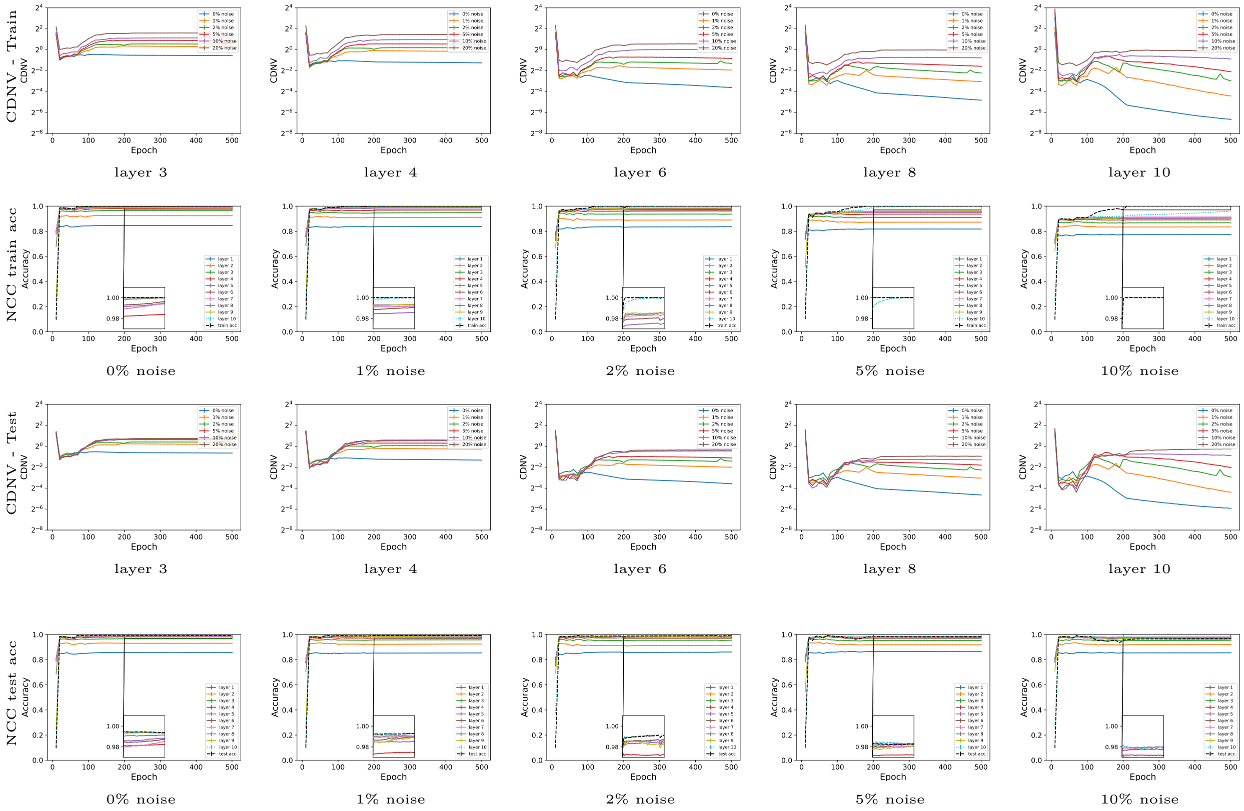
layer 4 layer 6 layer 8 layer 9 layer 10

0% noise 5% noise 10% noise 20% noise 30% noise

layer 4 layer 6 layer 8 layer 9 layer 10

0% noise 5% noise 10% noise 20% noise 30% noise

Figure 17: **Intermediate neural collapse of CONV-10-100 trained on Fashion MNIST with noisy labels.** See Fig. 3 in the main text for details.



layer 3 layer 4 layer 6 layer 8 layer 10

0% noise 1% noise 2% noise 5% noise 10% noise

layer 3 layer 4 layer 6 layer 8 layer 10

0% noise 1% noise 2% noise 5% noise 10% noise

Figure 18: **Intermediate neural collapse of CONV-10-50 trained on MNIST with partially corrupted labels.** See Fig. 14 for details.

B Proofs

Proposition 1. *Let $m \in \mathbb{N}$, $p \in (0, 1/2)$, $\alpha \in (0, 1)$ and $\epsilon \in (0, 1)$. Assume that the error of the learning algorithm is δ^1_m-uniform. Assume that $S_1, S_2 \sim P_B(m)$. Let $h^{\gamma}_{S_1}$ be the output of the learning algorithm given access to a dataset S_1 and initialization γ. Then,*

$$\begin{aligned} \mathbb{E}_{S_1}\mathbb{E}_{\gamma}[\mathrm{err}_P(h^{\gamma}_{S_1})] \;&\leq\; \mathbb{P}_{S_1,S_2,\tilde{Y}_2}\left[\mathbb{E}_{\gamma}[\mathscr{d}^{\epsilon}_{S_1}(h^{\gamma}_{S_1})] \;\geq\; \mathscr{d}^{\epsilon}_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2)\right] \\ &\quad + (1+\alpha)p + \delta^1_m + \delta^2_{m,p,\alpha}, \end{aligned} \tag{4}$$

where $\tilde{Y}_2 = \{\tilde{y}_i\}^m_{i=1}$ is uniformly selected to be a set of labels that disagrees with Y_2 on pm values.

Proof. Let $S_1 = \{(x^1_i, y^1_i)\}^m_{i=1}$ and $S_2 = \{(x^2_i, y^2_i)\}^m_{i=1}$ be two balanced datasets. Let $\epsilon > 0$, $p > 0$ and $q = (1 + \alpha)\ p$. Let $\tilde{Y}_2$ and $\hat{Y}_2$ be a uniformly selected set of labels that disagree with Y_2 on pm and qm randomly selected labels (resp.). We denote by $\tilde{S}_2$ and $\hat{S}_2$ the relabeling of S_2 with the labels in $\tilde{Y}_2$ and in $\hat{Y}_2$ (resp.). We define four different events,

$$\begin{aligned} A_1 \;&=\; \{(S_1, S_2, \tilde{Y}_2) \mid \exists\ q \geq (1+\alpha)\ p:\ \mathscr{d}^{\epsilon}_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2) > \mathbb{E}_{\hat{Y}_2}[\mathscr{d}^{\epsilon}_{\min}(\mathcal{G}, S_1 \cup \hat{S}_2)]\} \\ A_2 \;&=\; \{(S_1, S_2) \mid \text{the mistakes of } h^{\gamma}_{S_1} \text{ are not uniform over } S_2\} \\ A_3 \;&=\; \{(S_1, S_2, \tilde{Y}_2) \mid (S_1, S_2, \tilde{Y}_2) \notin A_1 \cup A_2 \text{ and } \mathbb{E}_{\gamma}[\mathscr{d}^{\epsilon}_{S_1}(h^{\gamma}_{S_1})] \;<\; \mathscr{d}^{\epsilon}_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2)\} \\ A_4 \;&=\; \{(S_1, S_2, \tilde{Y}_2) \mid (S_1, S_2, \tilde{Y}_2) \notin A_1 \cup A_2 \text{ and } \mathbb{E}_{\gamma}[\mathscr{d}^{\epsilon}_{S_1}(h^{\gamma}_{S_1})] \;\geq\; \mathscr{d}^{\epsilon}_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2)\} \\ B_1 \;&=\; \{(S_1, S_2, \tilde{Y}_2) \mid \mathbb{E}_{\gamma}[\mathscr{d}^{\epsilon}_{S_1}(h^{\gamma}_{S_1})] \;\geq\; \mathscr{d}^{\epsilon}_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2)\} \end{aligned} \tag{7}$$

By the law of total expectation

$$\begin{aligned} \mathbb{E}_{S_1}\mathbb{E}_{\gamma}[\mathrm{err}_P(h^{\gamma}_{S_1})] \;&=\; \mathbb{E}_{S_1,S_2}\mathbb{E}_{\gamma}[\mathrm{err}_{S_2}(h^{\gamma}_{S_1})] \\ &=\; \sum^4_{i=1} \mathbb{P}[A_i] \cdot \mathbb{E}_{S_1,S_2,\tilde{Y}_2}[\mathbb{E}_{\gamma}[\mathrm{err}_{S_2}(h^{\gamma}_{S_1})] \mid A_i] \\ &\leq\; \mathbb{P}[A_1] + \mathbb{P}[A_2] + \mathbb{E}_{S_1,S_2,\tilde{Y}_2}[\mathbb{E}_{\gamma}[\mathrm{err}_{S_2}(h^{\gamma}_{S_1})] \mid A_3] + \mathbb{P}[B_1], \end{aligned} \tag{8}$$

where the last inequality follows from $\mathrm{err}_{S_2}(h^{\gamma}_{S_1}) \leq 1$, $\mathbb{P}[A_3] \leq 1$ and $A_4 \subset B_1$.

We would like to upper bound each one of the above terms. First, we notice that since the mistakes of the network are δ^1_m-uniform, $\mathbb{P}[A_2] \leq \delta^1_m$. In addition, by definition $\mathbb{P}[A_1] \leq \delta^2_{m,p,\alpha}$.

As a next step, we upper bound $\mathbb{E}_{S_1,S_2,\tilde{Y}_2}[\mathbb{E}_{\gamma}[\mathrm{err}_{S_2}(h^{\gamma}_{S_1})] \mid A_3]$. Assume that $(S_1, S_2, \tilde{Y}_2) \in A_3$. Hence, $(S_1, S_2, \tilde{Y}_2) \notin A_1 \cup A_2$. Then, the mistakes of $h^{\gamma}_{S_1}$ over S_2 are uniformly distributed (with respect to the selection of γ). Assume by contradiction that $\mathrm{err}_{S_2}(h^{\gamma}_{S_1}) > (1 + \alpha)\ p$ with non-zero probability over the selection of γ. Then, since the mistakes of $h^{\gamma}_{S_1}$ over S_2 are uniformly distributed, $\mathrm{err}_{S_2}(h^{\gamma}_{S_1}) > (1 + \alpha)\ p$ for all initializations γ. Therefore, we have

$$\mathbb{E}_{\hat{Y}_2}[\mathscr{d}^{\epsilon}_{\min}(\mathcal{G}, S_1 \cup \hat{S}_2)] \;\leq\; \mathbb{E}_{\gamma}[\mathscr{d}^{\epsilon}_{S_1}(h^{\gamma}_{S_1})] \;<\; \mathscr{d}^{\epsilon}_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2),$$

where the first inequality follows from the definition of $\mathscr{d}^{\epsilon}_{\min}(\mathcal{G}, S_1 \cup \hat{S}_2)$ and the second one by the assumption that $(S_1, S_2, \tilde{Y}_2) \in A_3$. However, this inequality contradicts the fact that $(S_1, S_2, \tilde{Y}_2) \notin A_1$. Therefore, we conclude that in this case, $\mathbb{E}_{\gamma}[\mathrm{err}_{S_2}(h^{\gamma}_{S_1})] \leq (1 + \alpha)\ p$ and $\mathbb{E}_{S_1,S_2,\tilde{Y}_2}[\mathbb{E}_{\gamma}[\mathrm{err}_{S_2}(h^{\gamma}_{S_1})] \mid A_3] \leq (1 + \alpha)\ p$. □

Proposition 2. *Let $m \in \mathbb{N}$, $p \in (0, 1/2)$, $\alpha \in (0, 1)$ and $\epsilon \in (0, 1)$. Assume that the error of the learning algorithm is δ^1_m-uniform. Let $S_1, S_2, S^i_1, S^i_2 \sim P_B(m)$ (for $i \in [k]$). Let $\tilde{Y}^i_2 = \{\tilde{y}_i\}^m_{i=1}$ be a set of labels that disagrees with Y^i_2 on uniformly selected pm labels and $\tilde{S}^i_2$ is a relabeling of S_2 with the labels in $\tilde{Y}^i_2$. Let $h^{\gamma}_{S_1}$ be the output of the learning algorithm given access to a dataset S_1 and initialization γ. Then, with probability at least $1 - \delta$ over the selection of $\{(S^i_1, S^i_2, \tilde{Y}^i_2)\}^k_{i=1}$, we have*

$$\begin{aligned} \mathbb{E}_{S_1}\mathbb{E}_{\gamma}[\mathrm{err}_P(h^{\gamma}_{S_1})] \;&\leq\; \frac{1}{k}\sum^k_{i=1} \mathbb{I}\left[\mathbb{E}_{\gamma}[\mathscr{d}^{\epsilon}_{S^i_1}(h^{\gamma}_{S^i_1})] \;\geq\; \mathscr{d}^{\epsilon}_{\min}(\mathcal{G}, S^i_1 \cup \tilde{S}^i_2)\right] \\ &\quad + (1+\alpha)\ p + \delta^1_m + \delta^2_{m,p,\alpha} + \sqrt{\frac{\log(2/\delta)}{2k}}. \end{aligned}$$

Proof. By Prop. 1, we have

$$\begin{aligned}\mathbb{E}_{S_1}\mathbb{E}_{\gamma}[\mathrm{err}_P(h^{\gamma}_{S_1})] \;\leq\; & \mathbb{P}_{S_1,S_2,\tilde{Y}_2}\left[\mathbb{E}_{\gamma}[\mathscr{d}^{\epsilon}_{S_1}(h^{\gamma}_{S_1})] \;\geq\; \mathscr{d}^{\epsilon}_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2)\right] \\ & + (1+\alpha)\; p_m + \delta^1_m + \delta^2_{m,p,\alpha}\end{aligned}$$

We define i.i.d. random variables

$$V_i \;=\; \mathbb{I}\left[\mathbb{E}_{\gamma}[\mathscr{d}^{\epsilon}_{S^i_1}(h^{\gamma}_{S^i_1})] \;\geq\; \mathscr{d}^{\epsilon}_{\min}(\mathcal{G}, S^i_1 \cup \tilde{S}^i_2)\right]. \tag{9}$$

Therefore, we can rewrite,

$$\mathbb{P}_{S_1,S_2,\tilde{Y}_2}\left[\mathbb{E}_{\gamma}[\mathscr{d}^{\epsilon}_{S_1}(h^{\gamma}_{S_1})] \;\geq\; \mathscr{d}^{\epsilon}_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2)\right] \;=\; \mathbb{E}[V_1] \tag{10}$$

By Hoeffding's inequality,

$$\mathbb{P}\left[\left|k^{-1}\sum_{i=1}^{k} V_i - \mathbb{E}[V_1]\right| \;\geq\; \epsilon\right] \;\leq\; 2\exp(-2k\epsilon^2). \tag{11}$$

By choosing $\epsilon = \sqrt{\log(1/2\delta)/2k}$, we obtain that with probability at least $1-\delta$, we have

$$\mathbb{E}[V_1] \;\leq\; \frac{1}{k}\sum_{i=1}^{k} V_i + \sqrt{\log(1/2\delta)/2k}. \tag{12}$$

When combined with Prop. 1, we obtain the desired bound. □