
On the Implicit Bias Towards Depth Minimization in Deep Neural Networks

Tomer Galanti
CBMM
Massachusetts Institute of Technology
MA, USA
galanti@mit.edu

Liane Galanti
School of Computer Science
Tel Aviv University
Israel
lianegalanti@mail.tau.ac.il

Abstract

We study the implicit bias of gradient based training methods to favor low-depth solutions when training deep neural networks. Recent results in the literature suggest that penultimate layer representations learned by a classifier over multiple classes exhibit a clustering property, called neural collapse. We demonstrate empirically that neural collapse extends beyond the penultimate layer and emerges in intermediate layers as well. In this regards, we hypothesize and empirically show that gradient based methods are implicitly biased towards selecting neural networks of minimal depth for achieving this clustering property.

1 Introduction

Deep learning systems have steadily advanced the state of the art in a wide variety 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].

A central aspect that enables the success of these systems is the ability to train deep models instead of wide shallow ones [He et al., 2016]. Intuitively, a neural network is decomposed into hierarchical representations from raw data to high-level, more abstract features. While training deeper neural networks repetitively achieves superior performance against their shallow counterparts He et al. [2015], Wang et al. [2022], an understanding of the role of depth in representation learning is still lacking.

Multiple contributions Poggio et al. [2020], Safran et al. [2021], Eldan and Shamir [2016] studied the role of depth from the view point of approximation theory and expressivity. These works suggest that in certain cases it requires less parameters in order to approximate a given smooth target function using deeper networks. While these papers demonstrate superior approximation guarantees for deeper networks, it is typically possible to fit the training data (and test data) even by using a shallow fully-connected network. In addition, these papers measure the success of neural networks as the best approximation a given architecture provides for a given target function. Therefore, these papers do not take into account on the specific functionalities captured by different layers of the trained model.

As an attempt to understand the training dynamics of neural networks, another line of work considers the training dynamics of neural networks of very large widths [Jacot et al., 2018]. In [Lee et al., 2019] they showed that under certain conditions, training wide shallow neural networks with gradient decent converges to a solution of kernel regression with the neural tangent kernel (NTK). However, this result makes several unrealistic assumptions that give rise to training dynamics that abstain from learning representations. These assumptions include: large widths, incorporating a non-standard initialization procedure and the absence of batch normalization layers [Jacot et al., 2019].

Recently, Papayan et al. [2020] suggested a different perspective on understanding the representations learned by neural networks in a standard setting (e.g., with batch normalization, standard initialization, etc.) in the lens of neural collapse. Informally, neural collapse (NC) identifies training dynamics of deep networks for standard classification tasks, where the features of the penultimate layer associated with training samples belonging to the same class tend to concentrate around their means. Specifically, Papayan et al. [2020] observed that the ratio of the within-class variances and the distances between the class means tends to zero, especially at the terminal stage when training proceeds beyond perfectly fitting the training labels. They also noticed that asymptotically the class-means (centered at their global mean) are not only linearly separable, but are actually maximally distant and located on a sphere centered at the origin up to scaling (they form a simplex equiangular tight frame). Furthermore, it has also been observed that the features are *nearest class-center separable*, meaning that a nearest class-center classifier on top of the features is able to perfectly distinguish between the classes. In addition to that, it has been shown that the behavior of the last-layer classifier (operating on the features) converges to that of the nearest class-center decision rule. Since neural collapse deals with a setting in which the within-class variability of the penultimate layer is small, we can intuitively say that the classification is essentially already done at the penultimate layer.

Contributions. In this work we suggest a new perspective on understanding the role of depth in deep learning. We observe that *SGD training of deep neural networks exhibits an implicit bias that favors solutions of minimal depth for achieving nearest class-center classification.*

The central contributions in this work are:

- We empirically show that in contrast to original predictions, neural collapse does not necessarily happen at the stage of training when the network perfectly fits the data. Instead, we show that it emerges only if the network is sufficiently deep.
- We characterize and study the *effective depth* of neural networks that measures the lowest layer’s features that are NCC separable. We empirically show that when training a neural network, SGD favors solutions of small effective depths. Specifically, the feature embeddings of layers above a certain minimal depth of the neural network are NCC separable.
- We empirically show that the effective depth of neural networks increases when trained with extended portions of corrupted labels and we prove its connection with generalization.

Organization. The rest of the paper is organized as follows: Some additional related work is discussed in Section 1.1. The problem setting is introduced in Section 2. Neural collapse, the effective depths of neural networks and their relation with generalization are presented in Section 3. Experiments are presented in Section 4, and conclusions are drawn in Section 5.

1.1 Additional Related Work

Neural collapse and generalization. The relationship between neural collapse and generalization has been addressed multiple times in the literature. In [Galanti et al., 2022] they theoretically and empirically studied the conditions for neural collapse to generalize from train samples, to test samples and new classes and its implications on transfer learning. For instance, they showed that in the regime of neural collapse, if the number of training samples tends to infinity, we should expect to encounter neural collapse on the test samples as well. Inspired by that demonstrated the possibility of perfectly fitting random labels with neural networks despite their ability to generalize, Mixon et al. [2020] provided empirical evidence that neural collapse emerges when training the network with random labels. This raises the following question: *does neural collapse indicate whether the network generalizes?*

In this work, we make the following observations. First, we observe that the degree of neural collapse (on the training samples) generally improves when increasing the number of layers. Second, we extend the experiment of Mixon et al. [2020] we observe that the degree of neural collapse when training with the correct labels is lower than the neural collapse achieved when a subset of the labels are corrupted. Furthermore, we show that the minimal NCC depth increases when training neural networks with extended portions of corrupted labels. Therefore, we conclude that the presence of neural collapse itself only weakly indicates whether the network generalizes or not. Instead, we propose the minimal NCC depth as a stronger indicator whether generalization is evident or not.

Neural collapse in intermediate layers. While the original motivation in Papayan et al. [2020] was to study the variability collapse of the penultimate layer and the convergence of the network’s last layer to a nearest class-center classifier, it is tempting to understand whether this behaviour also extends below the network’s penultimate layer. For the purpose of studying the relationship between neural collapse and transferability, Galanti et al. [2022] investigated the emergence of neural collapse in intermediate layers. Specifically, they compared the degree of variability collapse in the second-to-last embedding layer of a residual network with the variability collapse in the penultimate layer. This paper observed that while we can see a similar clustering behaviour as in the penultimate layer, the extent is lower. Following that Hui et al. [2022] conducted an experiment showing a similar behaviour with MLPs. In Ben-Shaul and Dekel [2022] they consider the emergence of neural collapse intermediate layers of neural networks by looking at the nearest class-center classification error of various layers of the networks. However, none of these papers identified the minimal-depth principle that we characterize and study in this work.

2 Problem Setup

We consider the problem of training a model for a standard multi-class classification. Formally, the target task is defined by a distribution P over samples $(x, y) \in \mathcal{X} \times \mathcal{Y}_C$, where $\mathcal{X} \subset \mathbb{R}^d$ is the instance space, and \mathcal{Y}_C is a label space with cardinality C . To simplify the presentation, we use one-hot encoding for the label space, that is, the labels are represented by the unit vectors in \mathbb{R}^C , and $\mathcal{Y}_C = \{e_c : c = 1, \dots, C\}$ where $e_c \in \mathbb{R}^C$ is the c th standard unit vector in \mathbb{R}^C ; with a slight abuse of notation, sometimes we will also write $y = c$ instead of $y = e_c$. For a pair (x, y) distributed by P , we denote by P_c the class conditional distribution of x given $y = c$ (i.e., $P_c(\cdot) = \mathbb{P}[x \in \cdot \mid y = c]$).

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

$$L_P(h_W) = \mathbb{E}_{(x,y) \sim P}[\ell(h_W(x), y)], \quad (1)$$

where $\ell : \mathbb{R}^C \times \mathcal{Y}_C \rightarrow [0, \infty)$ is a non-negative loss function (e.g., L_2 or cross-entropy losses). For simplicity, sometimes we will omit writing W in the subscript of h .

We typically do not have direct access to the full population distribution P . Therefore, we typically aim to learn a the classifier, h , from 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 minimize the empirical risk

$$L_S(h) = \frac{1}{m} \sum_{i=1}^m \ell(h(x_i), y_i). \quad (2)$$

Finally, the performance of the trained model is evaluated using the train and test error rates, which are computed as follows: $err_S(h) = \sum_{i=1}^m \ell(h(x_i), y_i)$ and $err_P(h) = \mathbb{E}_{(x,y) \sim P}[\ell(h(x), y)]$.

Neural Networks. In this work, the classifier h is implemented as a neural network, decomposed into a set of parametric layers. Formally, we write $h := g^{L+1} \circ \dots \circ g^1 := g_{\theta_{L+1}}^{L+1} \circ \dots \circ g_{\theta_1}^1(x)$, where $g_{\theta_i}^i \in \{g' : \mathbb{R}^{p_i} \rightarrow \mathbb{R}^{p_{i+1}}\}$ are parametric functions with θ_i being vectors of real-values. For example, $g_{\theta_i}^i$ could be a standard layer $g_{\theta_i}^i(z) = \sigma(\theta_i \cdot z)$, a residual block $g_{\theta_i}^i(z) = z + \theta_i^2 \sigma(\theta_i^1 z)$ or a pooling layer. The last layer $g_{\theta_{L+1}}^{L+1}$ is simply a linear mapping. Here, σ is an element-wise activation function. We denote the i th layer of the neural network as $f_i = g_{\theta_i}^i \circ \dots \circ g_{\theta_1}^1(x)$, for $i \leq d$.

Optimization. In this work, we consider optimizing h using stochastic gradient decent (SGD). We aim at minimizing the regularized empirical risk $L_S^\lambda(h) = L_S(h) + \lambda \|W\|_2^2$ by applying SGD for a certain number of iterations. Namely, we initialize the weights W_0 of h using a standard initialization procedure (e.g., Kaiming He initialization) and at each iteration, we 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 the subset $\tilde{S} \subset S$ of size B is selected uniformly at random. Throughout the paper, we denote by $h_S = g_S^{L+1} \circ f_S$ the output of the learning algorithm.

Notations. Throughout the paper, we use the following notations. For an integer $k \geq 1$, $[k] = \{1, \dots, k\}$. For any real vector z , $\|z\|$ denotes its Euclidean norm. We denote by $\mu_u(Q) = \mathbb{E}_{x \sim Q}[u(x)]$ and by $\text{Var}_u(Q) = \mathbb{E}_{x \sim Q}[\|u(x) - \mu_u(Q)\|^2]$ the mean and variance of $u(x)$ for $x \sim Q$. For a finite set A , we denote by $U[A]$ the uniform distribution over A . We denote by $\mathbb{I} : \{\text{True}, \text{False}\} \rightarrow \{0, 1\}$ the indicator function. For a given distribution P over \mathcal{X} and a measurable function $f : \mathcal{X} \rightarrow \mathcal{X}'$, we denote the distribution of $f(x)$ by $f \circ P$. Let $U = \{y_i\}_{i=1}^m$ be a set of labels $y_i \in [C]$. We denote $D(U) = (p_1, \dots, p_C)$, with $p_c = \frac{1}{m} \sum_{i=1}^m \mathbb{I}[y_i = c]$.

3 Neural Collapse and Generalization

In this section we theoretically explore the relationship between neural collapse and generalization. We begin by formally introducing neural collapse, NCC separability along with the effective depth of neural networks. Then, we connect these notions with the test-time performance of neural networks.

3.1 Neural Collapse

As mentioned in Section 1, neural collapse considers 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. In this paper, we focus on the class-features variance collapse (NC1) and the nearest class-center classifier simplification (NC4) properties of neural collapse.

To evaluate NC1, we follow the evaluation process suggested by Galanti et al. [2022], that works with a slightly different variation of within-class variation collapse, which is related with the clusterability of the sample feature vectors. For a feature map $f : \mathbb{R}^d \rightarrow \mathbb{R}^p$ and two (class-conditional) distributions¹ Q_1, Q_2 over $\mathcal{X} \subset \mathbb{R}^d$, we define their *class-distance normalized variance* (CDNV) to be

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

The definition of Galanti et al. [2022] for *neural collapse* (at training) asserts that

$$\lim_{t \rightarrow \infty} \text{Avg}_{i \neq j \in [l]} [V_{f^{(t)}}(S_i, S_j)] = 0,$$

where $f^{(t)}$ is the value of the function f at iteration t of the training, focusing on f being the penultimate layer of the neural network $f = g^L \circ \dots \circ g^1$. As shown empirically in Galanti et al. [2022], this definition is essentially the same as that of Pappayan et al. [2020].

The nearest class-center classifier simplification property asserts that, during training, the feature embeddings in the penultimate layer become separable and classifier h itself converges to the ‘nearest class-center classifier’, \hat{h} . Formally, suppose we have a dataset $S = \{(x_i, y_i)\}_{i=1}^m = \cup_{c=1}^C S_c$ of samples and a feature map $f : \mathbb{R}^d \rightarrow \mathbb{R}^p$, we say that the features of f are NCC separable, if

$$\forall j \in [m] : \hat{h}(x_j) = \arg \min_{c \in [C]} \|f(x_j) - \mu_f(S_c)\| = y_j. \quad (3)$$

As additional measures of collapse of a given layer, we also use the NCC train and test error rates, $err_S(\hat{h}_i)$ and $err_P(\hat{h}_i)$, which are the error rates of $\hat{h}_i(x) = \arg \min_{c \in [C]} \|f_i(x) - \mu_{f_i}(S_c)\|$. As shown in [Galanti et al., 2022], the NCC error rate can be upper bounded in terms of the CDNV. However, the NCC error can be zero in cases where the CDNV would be larger than zero.

3.2 Effective Depths and Generalization

In this work we argue that neural networks trained for standard classification exhibit an implicit bias towards depth minimization. Namely, if by training a model $f_L = g^L \circ \dots \circ g^1$ of depth L for classification, the learned features f_L are NCC separable, then by training a neural network $f_{L+l} = g^{L+l} \circ \dots \circ g^1$ (with the same hyperparameters) we obtain that the features of the top l layers are NCC separable. Intuitively, we could correctly classify the samples already in the L ’th layer of f_{L+l} , and therefore, its depth is effectively upper bounded by L . The notion of effective depth is formally defined as follows.

¹The definition can be extended to finite sets $S_1, S_2 \subset \mathcal{X}$ by defining $V_f(S_1, S_2) = V_f(U[S_1], U[S_2])$.

Definition 1 (Effective depth). *Suppose we have a dataset $S = \cup_{c=1}^C S_c = \{(x_i, y_i)\}_{i=1}^m$, a neural network $f^L = g^L \circ \dots \circ g^1$ with $g^1 : \mathbb{R}^n \rightarrow \mathbb{R}^{p_2}$ and $g^i : \mathbb{R}^{p_{i+1}} \rightarrow \mathbb{R}^{p_i}$. The effective empirical depth $\mathcal{d}_S(f_L)$ of f_L is the minimal value $i \in [L]$, such that, $\hat{h}_i(x_j) = \arg \min_{c \in [C]} \|f_i(x_j) - \mu_{f_i}(S_c)\| = y_j$ for all $j \in [m]$.*

While our empirical observations in Sec. 4 demonstrate that the optimizer tends to learn neural networks of low-depths, it is not necessarily the lowest depth that allows NCC separability. As a next step, we define the *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 effective depth of a neural network and the minimal NCC depth is tightly related with generalization.

Definition 2 (Minimal NCC depth). *Suppose we have a dataset $S = \cup_{c=1}^C S_c = \{(x_i, y_i)\}_{i=1}^m$ neural network architecture $f^L = g^L \circ \dots \circ g^1$ with $g^1 : \mathbb{R}^n \rightarrow \mathbb{R}^{n_0}$ and $g^i \in \mathcal{G} \subset \{g' \mid g' : \mathbb{R}^{n_0} \rightarrow \mathbb{R}^{n_0}\}$ for all $i = 2, \dots, L$. The minimal NCC depth of the network is the minimal depth L for which there exists a feature map $f_L = g^L \circ \dots \circ g^1(x)$ that perfectly fit the data using a NCC classifier. We denote the minimal NCC depth by $\mathcal{d}_{\min}(\mathcal{G}, S)$.*

As a technicality, throughout the analysis we assume that if the output neural network h_{S_1} was trained on a given balanced dataset S_1 of size m , then, it is also reasonable to expect that the labels h_{S_1} produces on a new balanced set of samples $X_2 = \cup_{c=1}^C \{x_{ci}^2\}_{i=1}^{m_0}$ would also be fairly balanced. This notion is formally defined in the following definition.

Definition 3 ((ϵ, δ) -balancedness). *Let h_S be the output of our learning algorithm. Assume that h_S perfectly fits the dataset S . We say that the learning algorithm is (ϵ, δ) -balanced, if with probability at least $1 - \delta$ over the selection of $S_1 = \{(x_i, y_i)\}_{i=1}^m$, $S_2 = \{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^m \sim P_B(m)$, we have $\|D(\{h_{S_1}(\tilde{x}_i)\}_{i=1}^m) - (1/C, \dots, 1/C)\|_1 \leq \epsilon$.*

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

Proposition 1 (Comparative generalization). *Let $\epsilon > 0$, $\delta, p \in [0, 1]$. Assume that the learning algorithm is (ϵ, δ) -balanced. Assume that $S_1, S_2 \sim P_B(m)$. Let $h_{S_1} = g_{S_1}^{L+1} \circ f_{S_1}$ be the output of the learning algorithm given access to a dataset S_1 . Then,*

$$\mathbb{E}_{S_1}[\text{err}_P(h_{S_1})] \leq \mathbb{P} \left[\mathcal{d}_{S_1}(f_{S_1}) \geq \min_{\tilde{Y}_2 \in E_p^\epsilon(Y_2)} \mathcal{d}_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2) \right] + p + \delta, \quad (4)$$

where $E_p^\epsilon(Y_2)$ is the set of all $\tilde{Y}_2 = \{\tilde{y}_i\}_{i=1}^m$ that are ϵ -balanced and disagree with $Y_2 = \{y_i\}_{i=1}^m$ on at least p ratio of the labels.

The above proposition provides an upper bound on the expected test error of the classifier h_{S_1} selected by the learning algorithm using a balanced dataset S_1 of size m . This bound assumes that with access to a balanced dataset, S_1 , the learning algorithm returns a function h_{S_1} that behaves approximately like a uniform distribution over the samples X_2 . This is formally captured by the (ϵ, δ) -balancedness assumption we make regarding the learning algorithm.

In this case, our classifier's error is at most p if its effective depth is lower than the depth of any alternative network that perfectly fits S_1 , mistakes on p ratio of the samples in X_2 and generates fairly uniform labels on X_2 . In the other case, the expected error is, by definition, upper bounded by $\mathbb{P} \left[\mathcal{d}_{S_1}(h_{S_1}) \geq \min_{\tilde{Y}_2 \in E_p^\epsilon(Y_2)} \mathcal{d}_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2) \right]$. Combining the two cases gives us the bound in Eq. 4.

We note that $\min_{\tilde{Y}_2 \in E_p^\epsilon(Y_2)} \mathcal{d}_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2)$ essentially measures the minimal depth required in order to perfectly fit the labels $Y_1 \cup \tilde{Y}_2$, where p ratio of the labels in \tilde{Y}_2 are simply random. Hence, the first term in the bound measures the probability that the effective depth of f_{S_1} is smaller than the minimal depth required to fit a balanced dataset of size $2m$, where $p/2$ ratio of the samples are random.

In general, it is impossible to perfectly fit an increasing amount of random labels without increasing the size of the neural network. Therefore, we expect $\min_{\tilde{Y}_2 \in E_p^\epsilon(Y_2)} \mathcal{d}_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2)$ to increase as long as we increase p or m .

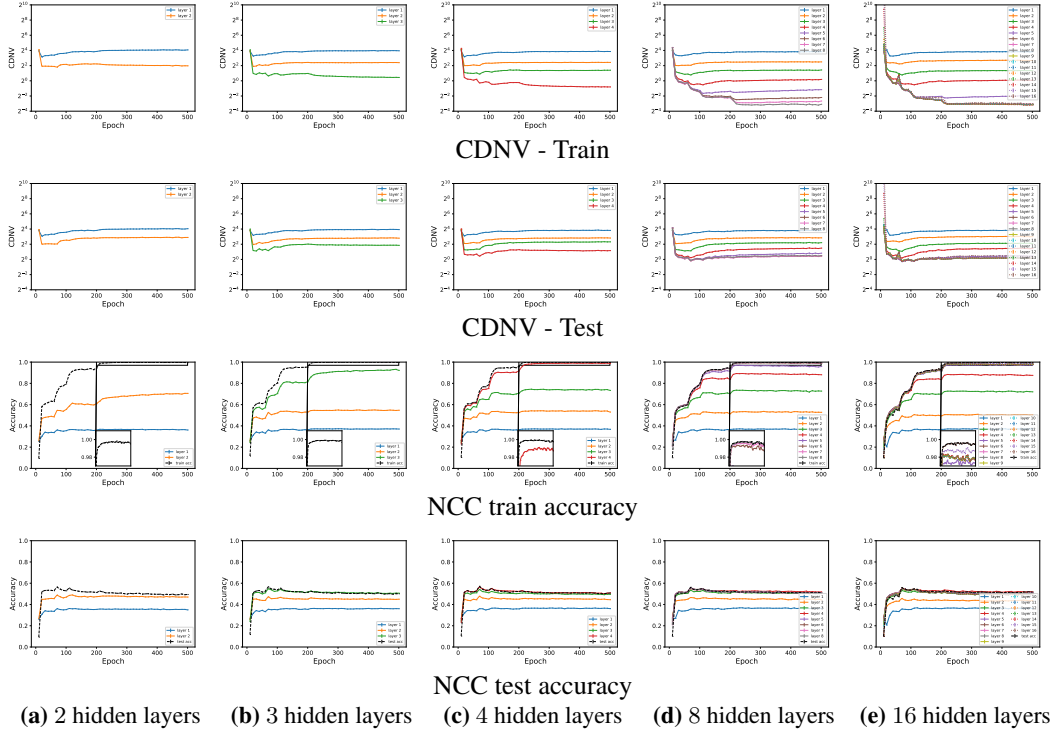


Figure 1: **Within-class feature variation collapse with MLP- d -100 trained on CIFAR10.** In (a-c) we plot the CDNV on the train data when varying the number of hidden layer in the network (plotted in lin-log scale). Each line stands for a different layer within the network. In (e) we plot the train accuracy rates of the various architecture.

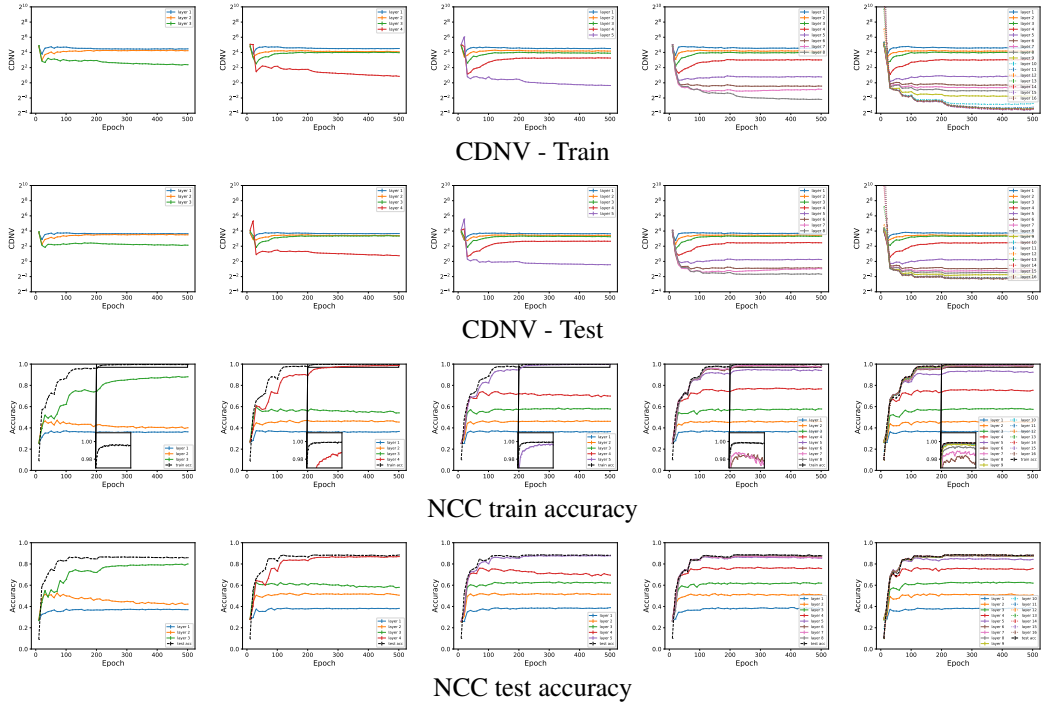
This generalization bound is fairly different than typical generalization bounds. Typically, the expected error is bounded by the sum between the train error and the ratio between a measure of complexity of the selected hypothesis (e.g., depending on the number of parameters, their norm, etc) and the square root of the number of training samples. Furthermore, measuring the generalization using a traditional generalization bounds would still result in a non-vacuous bound even if an implicit depth minimization is evident. That is because, the overall number of parameters of the network after replacing the top, redundant, layers with a nearest class-center classifier would still exceed the number of training samples.

4 Experiments

In this section we experimentally analyze the presence of neural collapse in the various layers of a trained network. In the first experiment, we validate the implicit bias of SGD to favor neural networks of minimal NCC depths. In the second experiment, we consider the effect of noisy labels on the extent of neural collapse, and specifically, on the minimal NCC depth. We show that NCC depth typically increases when extending the amount of noisy labels present in the data. Finally, we also conduct an experiment on the presence of neural collapse in a standard ResNet-18 classification problem to show that our observations happen in general settings as well.

4.1 Setup

Evaluation process. Following our problem setup, we consider k -class classification problems (e.g., CIFAR10, STL10) and train a multilayered neural network $h = g^d \circ f = g^L \circ \dots \circ g^1 : \mathbb{R}^n \rightarrow \mathbb{R}^C$ on some balanced training data $\mathcal{S} = \cup_{c=1}^C \mathcal{S}_c = \cup_{c=1}^C \{(x_{ci}, y_{ci})\}_{i=1}^{m_0}$. The model is trained using cross-entropy loss minimization between its logits and the one-hot encodings of the labels. Here, g^1, \dots, g^L are the various hidden layers of the network, where g^L is its top linear layer. As a second



(a) 3 hidden layers (b) 4 hidden layers (c) 5 hidden layers (d) 8 hidden layers (e) 16 hidden layers

Figure 2: **Within-class feature variation collapse with ConvNet- d -400 trained on CIFAR10.** In (a-c) we plot the CDNV on the train data when varying the number of hidden layer in the network (plotted in lin-log scale). Each line stands for a different layer within the network. In (e) we plot the train accuracy rates of the various architecture.

stage, the NCC accuracy of each sub-architecture $f_i = g^i \circ \dots \circ g^1(x)$ ($i \in [L]$) is evaluated as well. Throughout the experiments, we treat $> 99\%$ fitting of the training data as a (close to) perfect interpolation of the data.

Architectures and hyperparameters. In this work we consider three main architectures. The first architecture is a Multi-layered perceptron (MLP) whose depth is d and its width is w , denoted by MLP- d - w . Each layer consists of a linear layer, followed by batch normalization and ReLU. The second architecture is a simple convolutional neural network that begins with a stack of a 2×2 convolutional layer with stride 2, batch normalization, a convolution of the same structure, batch normalization and ReLU. Following that we have a set of d stacks of a 3×3 convolutional layer with stride 1 and padding 1, batch normalization and ReLU. The number of channels in each convolutional layer is w . The network is denoted by Conv- d - w . The third architecture is the standard ResNet-18 He et al. [2015].

The optimizations was carried out using SGD with batch size 128, momentum 0.9 and weight decay $5e-4$. We train h for 500 epochs.

Datasets. Throughout the experiments, we consider five different datasets: (i) MNIST; (ii) CIFAR10; (iii) CIFAR-100; (iv) STL-10; and (v) SVHN. For CIFAR variations we used random cropping, random horizontal flips and random rotations (by $15k$ degrees for k uniformly sampled from [24]). For all datasets we standardized the data.

4.2 Results

Minimal depth in neural networks. To study the bias towards learning solutions of minimal depth, we trained a set of fully connected neural networks with varying depths. In Figs. 1, 6 and 2 we plot the results of this experiment, for MLP- d -100, ConvNet- d -100 and ConvNet- d -400 networks. In each row we consider a different evaluation metric (the CDNV on the train and test data and the NCC classification accuracy on the train and test data) and in each column, we consider a neural network of

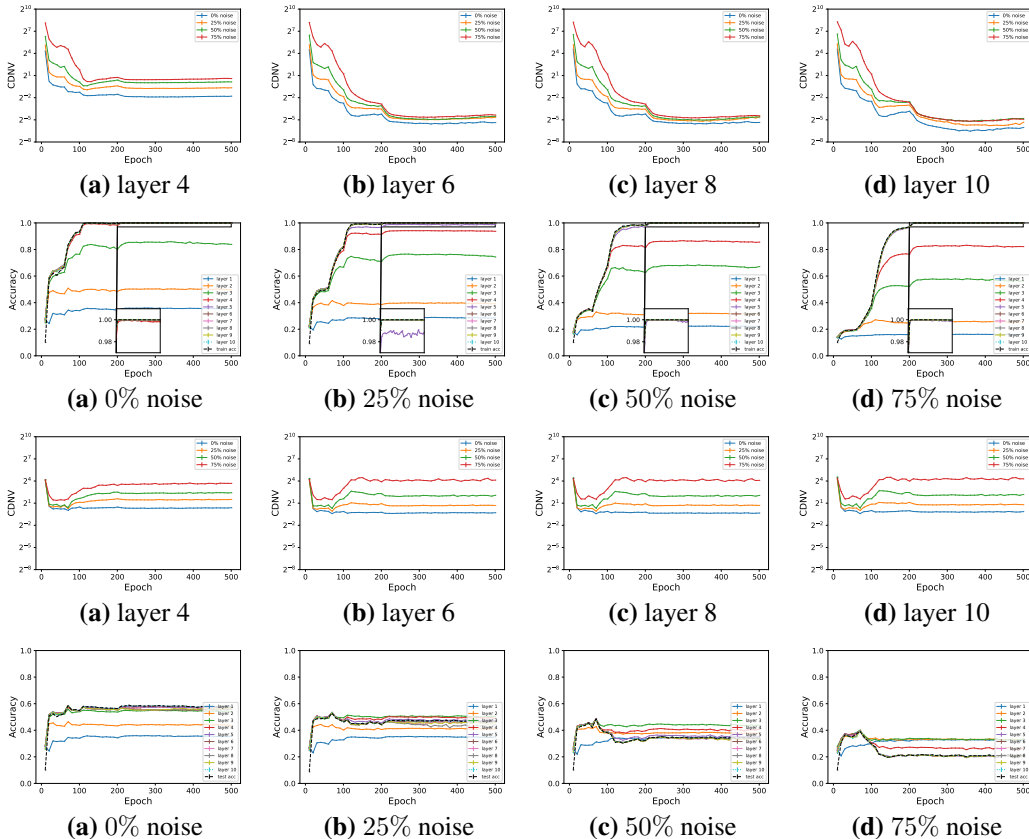


Figure 3: **Within-class feature variation collapse with MLP-10-500 trained on CIFAR10 with noisy labels.** In the first row we plot the CDNV on the train data for layers 4, 6, 8, 10 of networks trained with varying ratios of label noise (see legend). In the second row each figures reports the NCC accuracy rates of the various layers of a network trained with a certain amount of noisy labels (see titles). In the third and fourth rows we report the same experiments, but for the test data.

a different depth $d = 2, 3, 4, 8, 16$. The i 'th line stands for the CDNV at train time of the i 'th hidden layer of the neural network. As can be seen from the first row of Fig. 1, for the networks with 8 and 16 hidden layers, the fifth and higher layers enjoy neural collapse. Therefore, these networks are effectively of depth 4, which is exactly the minimal depth that allows $> 99\%$ fitting of the training data as observed in the third row of Fig. 1. The same can be observed with Fig. 2 for convolutional networks. In this case, the minimal depth is 5 instead of 4.

Emergence of neural collapse and overparameterization. Originally, neural collapse has been associated with the terminal stages of training, when the network perfectly fits the training data. As can be seen in Figs. 1(3,a) and 2(3,a), in certain cases, a relatively shallow network is sufficiently overparameterized in order to perfectly fit the data. In these cases, we do not necessarily achieve an emergence of neural collapse. As can be seen from the experiments in Figs. 1 and 2, neural collapse emerges only when it is possible to perfectly fit the training data with a nearest class-center classifier as the top layer.

Neural collapse with random labels. We follow Mixon et al. [2020] and investigate the emergence of neural collapse in the presence of random labels. In Mixon et al. [2020] they showed the neural collapse tends to happen on the training data even in the case when 100% of the training labels are selected uniformly at random from $[C]$. In this experiment we intend to compare the *degree* of collapse in the various layers of a neural network that is being trained with different portions of random labels. We train MLP-10-500 for CIFAR10 classification with 0%, 25%, 50% and 75% of the labels being replaced with random labels. We compare the degree of redundancy of the various layers of the networks as a function of the amount of noise in the data. The results are summarized in

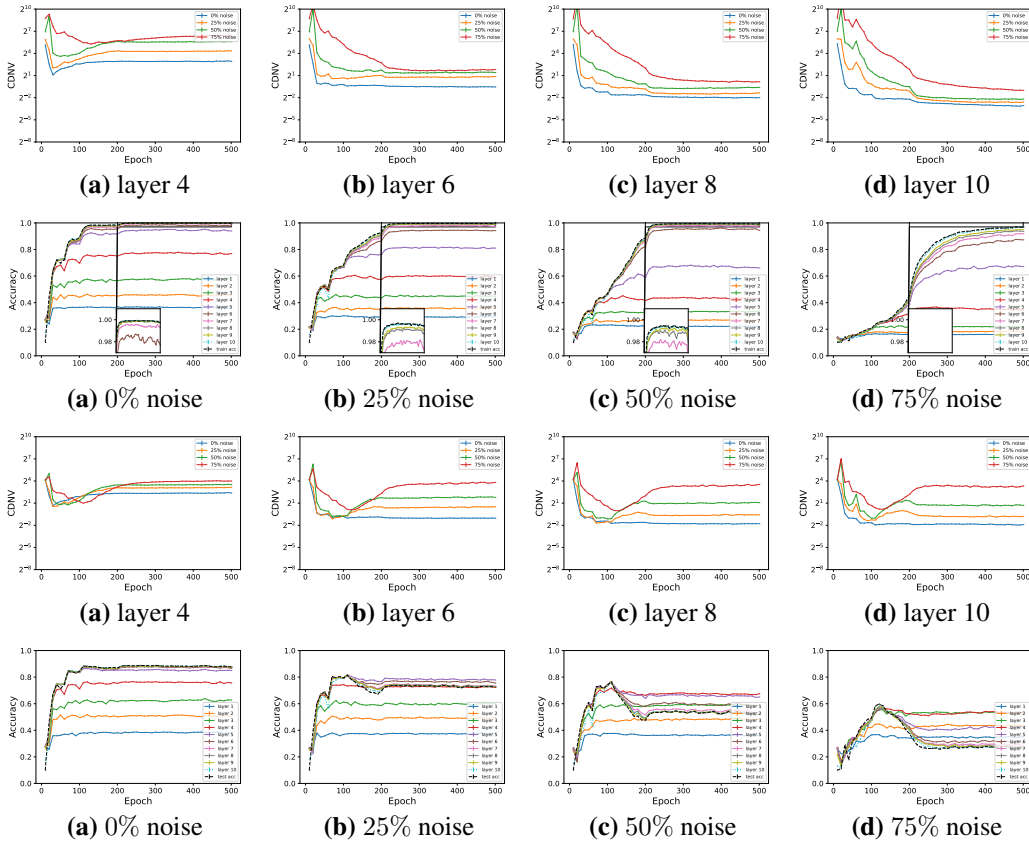


Figure 4: **Within-class feature variation collapse with ConvNet-10-500 trained on CIFAR10 with noisy labels.** In the first row we plot the CDNV on the train data for layers 4, 6, 8, 10 of networks trained with varying ratios of label noise (see legend). In the second row each figures reports the NCC accuracy rates of the various layers of a network trained with a certain amount of noisy labels (see titles). In the third and fourth rows we report the same experiments, but for the test data.

Fig. 3. In the first row we plot the CDNV of the i th layer of the various models on the train data for different layers in the networks, with $i = 4, 6, 8, 10$. In the second row we plot the NCC accuracy rates of the various layers of the models, when trained with different amounts of random labels. We also compare the performance with the train accuracy rate of the full model. In the third and fourth rows we plot exactly the same as the first two rows, but for the test data. The experiment is repeated for ConvNet-10-500 trained on CIFAR10 and the results are reported in Fig. 4.

As can be seen, we get a higher degree of neural collapse on each one of the layers. Specifically, we observe that within the fourth layer we obtain a CDNV of ≈ 0.28 for the network trained with 0% random labels, ≈ 0.6 with 25% random labels and CDNV above 1 for the others. Interestingly, we also observe that when training with noisy labels, the fourth layer’s NCC accuracy rate is lower than 100%, despite the fact that the network itself achieved 100% accuracy. In contrast, for a network trained with zero noisy labels, both the training accuracy and the NCC accuracy of the fourth layer are 100%. In fact, we can see that the NCC accuracy rate of the fourth layer decreases when increasing the amount of noisy labels. Therefore, we conclude that the NCC accuracy rate of the minimal depth is more predictive of the actual test performance than the training accuracy rate.

Implicit depth regularization in deep networks. As an additional experiment, we study the presence of neural collapse with standard ResNet-18 training on multiple datasets. During training, we evaluate the CDNV on the train and test data along with the layer’s performance (see ‘Method’). The CDNV is computed over each layer following a group of residual blocks. To train each model, we used learning rate scheduling with an initial learning rate 0.1, decayed three times at epochs 60,

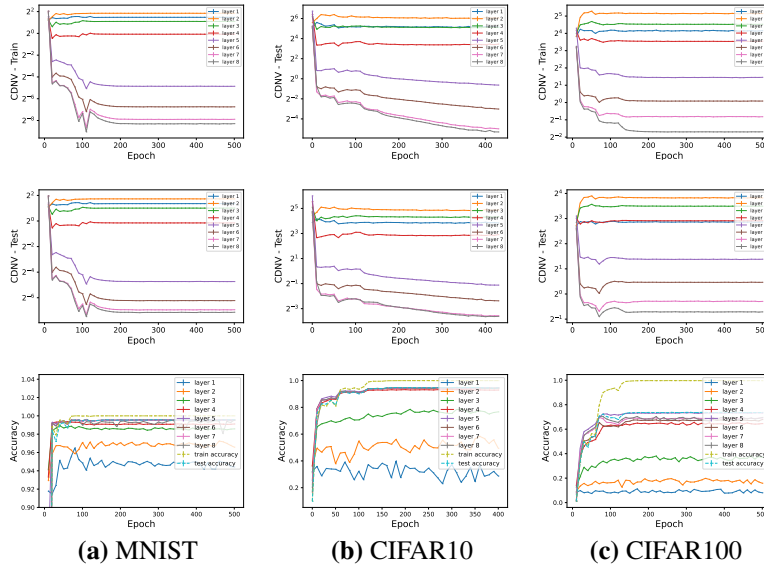


Figure 5: **Within-class feature variation collapse with ResNet-18.** We plot (row 1) the CDNV over the train data, (row 2) the CDNV over the test data and (row 3) the embedding performances (see 'Method') along with the train and test accuracy rates. In each column we present the results on a different datasets.

120, and 160. The results are summarized in Fig. 5. As can be seen, the few top-most evaluated layers of the neural network tend to collapse. In addition, in Fig. 5(c), we observe that collapsed layers tend to be redundant in the sense that their performance already matches that of the full network. In that sense, we argue that SGD implicitly prunes the top layers of the trained neural network, since it tends to select weights for which the top layers are replaceable by a linear classifier.

5 Conclusions

Training deep neural networks is a successful approach for a variety of learning problems. However, the role of depth in deep learning and the functionalities learned by each layer are not very clear. In this paper we presented a new perspective on this problem by studying the emergence of neural collapse in the various layers during training.

While Papayan et al. [2020] suggested that neural collapse is a property that emerges in the penultimate layer of an overparameterized neural network at the stage where perfect fitting of the data is achieved, we suggest several extensions to this observation. First, we show that neural collapse does not necessarily emerge when perfect fitting to the data is achieved. In fact, we show that neural collapse emerges only when the network is sufficiently deep.

In addition, we empirically showed that the within-class variance collapse and NCC separability tend to emerge within the intermediate layers of a neural network and not just in the penultimate layer. Furthermore, we observe that if a network is able to perfectly fit the training data with a nearest class-center classifier as its top layer, then, any network with additional layers would exhibit the same degree of collapse.

Acknowledgements

This material is based upon work supported by the Center for Minds, Brains and Machines (CBMM), funded by NSF STC award CCF-1231216.

The authors would like to thank Tomaso Poggio, András György, Andrzej Banburski, Ido Ben-Shaul and X. Y. Han for illuminating discussions during the preparation of this manuscript.

References

- K. Arulkumaran, A. Cully, and J. Togelius. Alphastar: An evolutionary computation perspective, 2019. URL <http://arxiv.org/abs/1902.01724>. cite arxiv:1902.01724.
- I. Ben-Shaul and S. Dekel. Nearest class-center simplification through intermediate layers. *CoRR*, abs/2201.08924, 2022. URL <https://arxiv.org/abs/2201.08924>.
- T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.
- M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba. Evaluating large language models trained on code, 2021.
- J. Devlin, M.-W. Chang, K. Lee, and K. 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.
- R. Eldan and O. Shamir. The power of depth for feedforward neural networks, 2016.
- T. Galanti, A. György, and M. Hutter. On the role of neural collapse in transfer learning. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=SwIp410B6aQ>.
- K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.
- K. He et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet. In *ICCV*, 2015.
- L. Hui, M. Belkin, and P. Nakkiran. Limitations of neural collapse for understanding generalization in deep learning. *arXiv preprint arXiv:2202.08384*, 2022.
- A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS, page 8580–8589, Red Hook, NY, USA, 2018. Curran Associates Inc.
- A. Jacot, F. Gabriel, and C. Hongler. Freeze and chaos for dnns: an ntk view of batch normalization, checkerboard and boundary effects. *CoRR*, abs/1907.05715, 2019. URL <http://arxiv.org/abs/1907.05715>.
- J. Lee, L. Xiao, S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in Neural Information Processing Systems 32*, pages 8572–8583. Curran Associates, Inc., 2019.
- D. G. Mixon, H. Parshall, and J. Pi. Neural collapse with unconstrained features, 2020.

- V. Pappayan, X. Y. Han, and D. 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.
- T. Poggio, A. Banburski, and Q. Liao. Theoretical issues in deep networks. *Proceedings of the National Academy of Sciences*, 2020.
- I. Safran, R. Eldan, and O. Shamir. Depth separations in neural networks: What is actually being separated?, 2021.
- D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. 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.
- Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.
- H. Wang, S. Ma, L. Dong, S. Huang, D. Zhang, and F. Wei. Deepnet: Scaling transformers to 1,000 layers. *arXiv preprint arXiv:2203.00555*, 2022.
- X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer. Scaling vision transformers, 2021.

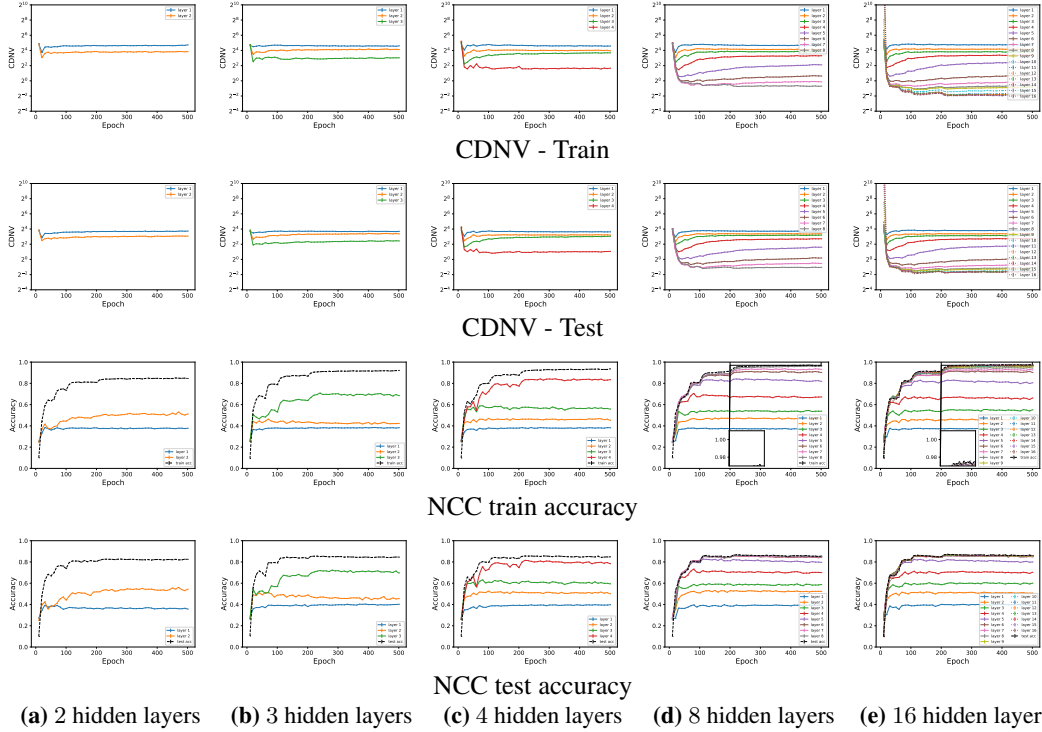


Figure 6: **Within-class feature variation collapse with ConvNet- d -100 trained on CIFAR10.** In (a-c) we plot the CDNV on the train data when varying the number of hidden layer in the network (plotted in lin-log scale). Each line stands for a different layer within the network. In (e) we plot the train accuracy rates of the various architecture.

A Additional Experiments

In this section we provide some additional experiments.

B Proofs

Proposition 1 (Comparative generalization). *Let $\epsilon > 0$, $\delta, p \in [0, 1]$. Assume that the learning algorithm is (ϵ, δ) -balanced. Assume that $S_1, S_2 \sim P_B(m)$. Let $h_{S_1} = g_{S_1}^{L+1} \circ f_{S_1}$ be the output of the learning algorithm given access to a dataset S_1 . Then,*

$$\mathbb{E}_{S_1}[\text{err}_P(h_{S_1})] \leq \mathbb{P} \left[\mathcal{d}_{S_1}(f_{S_1}) \geq \min_{\tilde{Y}_2 \in E_p^\epsilon(Y_2)} \mathcal{d}_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2) \right] + p + \delta, \quad (4)$$

where $E_p^\epsilon(Y_2)$ is the set of all $\tilde{Y}_2 = \{\tilde{y}_i\}_{i=1}^m$ that are ϵ -balanced and disagree with $Y_2 = \{y_i\}_{i=1}^m$ on at least p ratio of the labels.

Proof. Let $S_1 = \{(x_i^1, y_i^1)\}_{i=1}^m$ and $S_2 = \{(x_i^2, y_i^2)\}_{i=1}^m$ be two balanced datasets. Let \tilde{Y}_2 be an ϵ -balanced set of labels that disagrees with Y_2 on at least p labels. We define four different events:

$$\begin{aligned} A_1 &= \{S_1, S_2 \mid \{h_{S_1}(\tilde{x}_i^2)\}_{i=1}^m \text{ is not } \epsilon\text{-balanced}\} \\ A_2 &= \{S_1, S_2 \mid \{h_{S_1}(\tilde{x}_i^2)\}_{i=1}^m \text{ is } \epsilon\text{-balanced and } \mathcal{d}(h_{S_1}) < \min_{\tilde{Y}_2} \mathcal{d}_{\min}(\mathcal{F}, S_1 \cup \tilde{S}_2)\} \\ A_3 &= \{S_1, S_2 \mid \{h_{S_1}(\tilde{x}_i^2)\}_{i=1}^m \text{ is } \epsilon\text{-balanced and } \mathcal{d}(h_{S_1}) \geq \min_{\tilde{Y}_2} \mathcal{d}_{\min}(\mathcal{F}, S_1 \cup \tilde{S}_2)\} \\ B_1 &= \{S_1, S_2 \mid \mathcal{d}(h_{S_1}) \geq \min_{\tilde{Y}_2} \mathcal{d}_{\min}(\mathcal{F}, S_1 \cup \tilde{S}_2)\} \end{aligned} \quad (5)$$

By the law of total expectation

$$\begin{aligned}
\mathbb{E}_{S_1}[\text{err}_P(h_{S_1})] &= \mathbb{E}_{S_1, S_2}[\text{err}_{S_2}(h_{S_1})] \\
&= \sum_{i=1}^3 \mathbb{P}[A_i] \cdot \mathbb{E}_{S_1, S_2}[\text{err}_{S_2}(h_{S_1}) \mid A_i] \\
&\leq \mathbb{P}[A_1] + \mathbb{E}_{S_1, S_2}[\text{err}_{S_2}(h_{S_1}) \mid A_2] + \mathbb{P}[B_1],
\end{aligned} \tag{6}$$

where the last inequality follows from $\text{err}_{S_2}(h_{S_1}) \leq 1$, $\mathbb{P}[A_2] \leq 1$ and $A_3 \subset B_1$. We consider that if $\{h_{S_1}(\tilde{x}_i^2)\}_{i=1}^m$ is ϵ -balanced and $d(h_{S_1}) < \min_{\tilde{Y}_2} d_{\min}(\mathcal{F}, S_1 \cup \tilde{S}_2)$, then $h_{S_1}(x_j^2) \neq y_j^2$ on at most p ratio of the samples in S_2 . Therefore, $\mathbb{E}_{S_1, S_2}[\text{err}_{S_2}(h_{S_1}) \mid A_2] \leq p$. In addition, since the learning algorithm is (ϵ, δ) -balanced, $\mathbb{P}[A_1] \leq \delta$.

□