On the Implicit Bias Towards Depth Minimization in Deep Neural Networks

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

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

Ido Ben-Shaul  
Department of Applied Mathematics  
Tel Aviv University, Israel  
& eBay Research  
ido.benshaul@gmail.com

Abstract

Recent results in the literature suggest that the penultimate layer representations of neural networks that are trained for classification exhibit a clustering property called neural collapse (NC). We study the implicit bias of stochastic gradient descent (SGD) in favor of low-depth solutions when training deep neural networks. We characterize a notion of effective depth that measures the minimal layer that enjoys neural collapse. Furthermore, we hypothesize and empirically show that SGD implicitly selects neural networks of small effective depths.

Secondly, while neural collapse emerges even when generalization should be impossible - we argue that the rate of collapse in the intermediate layers is more sensitive, and is closely intertwined with generalization. We derive a generalization bound based on comparing the effective depth of the network with the minimal depth required to fit partially corrupted labels. Remarkably, this bound provides non-trivial estimations of the test performance. Finally, we empirically show that the effective depth of a trained neural network monotonically increases when training with extended portions of random labels.

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 [23, 26], language processing [5, 3], open-ended environments [22, 1], to coding [4].

A central aspect that enables the success of these systems is the ability to train deep models instead of shallow ones [11]. Intuitively, a neural network is decomposed into hierarchical representations from raw data to high-level, more abstract features. While training deeper networks repetitively achieves superior performance against their shallow counterparts [12] [25], an understanding of the role of depth in representation learning is still lacking.

Multiple contributions (e.g., [18, 20, 6]) studied the role of depth from the viewpoint of approximation theory. These works suggest that in certain cases it requires fewer parameters 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 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 the specific functionalities captured by different layers of the trained model.

Recently, [17] [10] suggested a different perspective for understanding the representations learned by neural networks in a conventional classification setting (e.g., trained 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, [17] observed that the ratio of the within-class variances and the distances between the class means is implicitly minimized, 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 (approximately) 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 (NCC) separable, meaning that a nearest class-center classifier on top of the features can perfectly distinguish between the classes and the last-layer classifier (operating on the features) converges to that of the nearest class-center decision rule. Following this work several papers analyzed the emergence of neural collapse from a theoretical standpoint (e.g., [27, 19, 14, 8, 7]).

While neural collapse is an intriguing phenomenon, its specific role in deep learning and its potential relationship with generalization is still unclear.

Even though neural collapse was originally introduced as a top-layer phenomenon, several papers [9, 2, 13] studied whether this behavior extends below the network’s penultimate layer. These papers provide preliminary results suggesting that neural collapse emerges at intermediate layers but to weaker degrees. Although intuitively intermediate neural collapse may seem linked to generalization, it is still unclear if (and how) these two properties are connected.

A separate line of work [9] studied the conditions for class features variation collapse generalizes from the train samples, to test samples and new classes and its connection with transfer learning. For instance, they showed that in the regime of neural collapse, if the number of training samples and classes tends to infinity, we should also encounter neural collapse on the test samples and new unseen classes. Following that, [13] further studied whether neural collapse generalizes to test samples.

Regardless of whether neural collapse generalizes to the test data, in this work, we focus on the following (independent) question: is neural collapse a good indication of whether the network generalizes well? As an argument against it, [15] provided empirical evidence that neural collapse emerges even when training the network with random labels. Therefore, the presence of neural collapse cannot indicate whether the network generalizes or not. However, this experiment does not invalidate the possibility of an indirect relationship between neural collapse and generalization. We argue that the rate of collapse in the intermediate layers is more sensitive, and may be closely intertwined with generalization.

Contributions. In this work, we suggest a new angle in understanding the role of depth and generalization in deep learning. The main contributions in this work are:

• We study the relationship between neural collapse and depth. Specifically, we empirically show that neural collapse emerges exclusively when the network is sufficiently deep and generally improves when increasing the network’s depth.

• We characterize and study the ‘effective depth’ of neural networks that measures the lowest layer’s features that are NCC separable. We provide robust experimentation showing that when training a neural network, SGD favors solutions of small effective depths, i.e., the feature embeddings of layers above a certain minimal depth of the neural network are NCC separable.

• We derive a generalization bound based on comparing the effective depth of the network with the minimal depth required to fit partially corrupted labels. This bound provides non-trivial estimations of the test performance. Furthermore, we empirically show that the effective depth of neural networks increases when trained with extended amounts of corrupted labels.

2
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 $\mathcal{Y}_C := \{e_c : c = 1, \ldots, C\}$ 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 $y = e$ 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., $P_c(\cdot) := \mathbb{P}[x \in \cdot | y = c]$).

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). 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 $S = \{(x_i, y_i)\}_{i=1}^{m} = \cup_{c=1}^{C} S_c = \cup_{c=1}^{C} [x_{ci}, y_{ci}]_{i=1}^{m_c} \sim P_B(m)$ 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

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

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, which are computed as follows: $\text{err}_S(h_W) := \sum_{i=1}^{m} \mathbb{I}[\arg\max_c h_W(x_i) \neq y_i]$ and $\text{err}_P(h_W) := \mathbb{E}_{(x, y) \sim P}[\mathbb{I}[\arg\max_c h_W(x) \neq y]]$.

Neural networks. In this work, the classifier $h_W$ is a neural network, decomposed into a set of parametric layers. Formally, we write $h_W := e_W \circ f_{W_L} := e_W \circ g_L^{W_L} \circ \cdots \circ g_1^{W_1}$, where $g_i^{W_i} \in \{g' : \mathbb{R}^{p_i} \to \mathbb{R}^{p_{i+1}}\}$ are parametric functions and $e_W \in \{e' : \mathbb{R}^{L+1} \to \mathbb{R}^C\}$ is a linear function. For example, $g_i^{W_i}$ could be a standard linear or convolutional layer, a residual block or a pooling layer. Here, $\sigma$ is an element-wise ReLU activation function. We denote the $i$th layer of the neural network by $f^i = g^i \circ \cdots \circ g_1$. With a slight abuse of notations, sometimes we will omit specifying the sub-scripted weights, e.g., $h := h_W$ and $f^i := f_{W_i}$.

In this work, we give special attention to the following architectures. The first architecture is a convolutional network, denoted by CONV-L-$H$. The network starts with a stack of a $2 \times 2$ convolutional layer with stride 2, batch normalization, a convolution of the same structure, batch normalization, and ReLU. Following that we have a set of $L$ stacks of $3 \times 3$ convolutional layers with $H$ channels, stride 1 and padding 1, batch normalization, and ReLU. The last layer is linear. The output tensors of these layers share the same shape as their input’s shape. The second architecture is an MLP, denoted by MLP-L-$H$ consisting of $L$ hidden layers, where each layer contains a linear layer of width $H$, followed by batch normalization and ReLU. The last layer is linear.

Optimization. We optimize our models to minimize the regularized empirical risk $L_S^\lambda(h) = L_S(h) + \lambda \|W\|^2$ by applying SGD for a certain number of iterations $T$ with coefficient $\lambda > 0$. Specifically, we initialize the weights $W_0 = \gamma$ of $h$ using a standard initialization procedure and at each iteration, we update $W_{t+1} \leftarrow W_t - \mu_t \nabla_W L_S(h_t)$, where $\mu_t > 0$ is the learning rate at the $t$th iteration and the subset $\hat{S} \subset S$ of size $B$ is selected uniformly at random. Throughout the paper, we denote by $h_S^\gamma$ the output of the learning algorithm starting from the initialization $W_0 = \gamma$. When $\gamma$ is irrelevant or obvious from context, we will simply write $h_S = h_S \circ e_S \circ f_S$.

Notations. Throughout the paper, we use the following notations. For an integer $k \geq 1$, $\{k\} := \{1, \ldots, k\}$. For any real vector $z \in \mathbb{R}^n$, $\|z\| := \sqrt{\sum_{i=1}^{n} z_i^2}$ 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 $I : \{\text{True, False}\} \to \{0, 1\}$ the indicator function. For a given distribution $P$ over $\mathcal{X}$ and a
measurable function \( f : \mathcal{X} \to \mathcal{X'} \), 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 \( D(U) := (p_1, \ldots, p_C) \), 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 start by formally introducing neural collapse, NCC separability, and 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 variations of the class-features variance collapse (NC1) and the nearest class-center classifier simplification (NC4) properties of neural collapse.

To evaluate NC1, we follow the process suggested by [9], which is a simplified version of the original approach of [17]. For a feature map \( f : \mathbb{R}^d \to \mathbb{R}^p \) and two (class-conditional) distributions \( Q_1, Q_2 \) over \( \mathcal{X} \subset \mathbb{R}^d \), we define their class-distance normalized variance (CDNV) to be

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

Essentially, this quantity measures to what extent the feature vectors of samples from \( Q_1 \) and \( Q_2 \) are clustered in space. The definition of [9] for neural collapse (at training) asserts that

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

where \( f(t) \) is the penultimate layer \( f = g^p \circ \cdots \circ g^1 \) of the neural network \( h = e \circ f \) at iteration \( t \) of training. As shown in [9], this definition is essentially the same as that of [17].

The nearest class-center classifier simplification property (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’, \( \hat{h} \) (see Eq. 3). Formally, suppose we have a dataset \( S = \bigcup_{c=1}^C S_c \) of samples and a mapping \( f : \mathbb{R}^d \to \mathbb{R}^p \), we say that the features of \( f \) are NCC separable, if

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

As additional measures of collapse of a given layer, we also use the NCC train and test error rates, \( \text{err}_t(\hat{h}) \) and \( \text{err}_t(\hat{h}) \). In a sense, NCC separability is a weaker notion of collapse. As shown by [9], 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 is larger than zero.

#### 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 (e.g., CONV-L-H or MLP-L-H). 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 CONV-L-H or MLP-L-H of any depth \( L \geq L_0 \) for cross-entropy minimization on \( S \) using SGD with weight decay, the learned features \( f^l \) become NCC separable for all \( l \in \{L_0, \ldots, L \} \).

In particular, 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 is formally defined as follows.

\[\text{depth} \] The definition can be extended to finite sets \( S_1, S_2 \subset \mathcal{X} \) by defining \( V_f(S_1, S_2) = V_f(U[S_1], U[S_2]) \).
Definition 1 (Empirical effective depth). Suppose we have a dataset $S$, a neural network $h = e \circ g^1 \circ \cdots \circ g^l$ with $g^i : \mathbb{R}^n \to \mathbb{R}^p_{i}$ and linear classifier $e : \mathbb{R}^{p_L+1} \to \mathbb{R}$. Let $h_i(x) = \arg \min_{c \in [C]} \|f_i(x) - \mu_{f_i}(S_i)\|$. The $\epsilon$-empirical effective depth $\delta^*_{\epsilon}(h)$ of $h$ is the minimal value $i \in [L]$, such that, $\text{err}_{\gamma}(h_i) \leq \epsilon$ (and $\delta^*_{\epsilon}(h) = L$ if such $i \in [L]$ is non-existent).

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 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 connected with generalization.

Definition 2 (Minimal NCC depth). Suppose we have a dataset $S = \bigcup_{i=1}^{C} S_i$ and a neural network architecture $f_i^L = g^L \circ \cdots \circ g^1$ with $g^i : \mathbb{R}^n \to \mathbb{R}^{n_{i+1}}$ and $g^i \in \mathcal{G} \subset \{g' : g' : \mathbb{R}^{n_{i+1}} \to \mathbb{R}^{n_{i+1}}\}$ for all $i = 2, \ldots, L$. The minimal NCC depth of $\mathcal{G}$ is the minimal depth $L$ for which there exist parameters $W = \{W_i\}_{i=1}^{L}$, such that, $f_i^L = g^L_{W_{2i}} \circ \cdots \circ g^1_{W_1}$ satisfies $\text{err}_{\gamma}(h) \leq \epsilon$, where $h(x) := \arg \min_{c \in [C]} \|f_i(x) - \mu_{f_i}(S_i)\|$. We denote the minimal NCC depth by $\delta^*_{\min}(\mathcal{G}, S)$.

To study the performance of a given model, we consider the following setup. Let $S_1 = \{(x_i^1, y_i^1)\}_{i=1}^{m_1}$ and $S_2 = \{(x_i^2, y_i^2)\}_{i=1}^{m_2}$ be two balanced datasets. We think of them as two splits of the training dataset $S$. We assume that the classifier $h_{S_1}^\gamma$ is trained on $S_1$ and we use $S_2$ to evaluate its performance.

We denote by $X_1 = \{x_i^1\}_{i=1}^{m_1}$ and $Y_2 = \{y_i^2\}_{i=1}^{m_2}$ the instances and labels in $S_1$. To formally state our bound, we make two technical assumptions. The first is that the misclassified labels that $h_{S_1}^\gamma$ produces over the samples $X_2 = \bigcup_{i=1}^{C} (x_i^2, y_i^2)_{i=1}^{m_1}$ are distributed uniformly.

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

The above definition provides two conditions regarding the learning algorithm. It assumes that with a high probability (over the selection of $S_1$, $S_2$), $h_{S_1}^\gamma$ makes a constant number of mistakes on $X_2$ across all initializations $\gamma$. 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 error typically has a small variance and the mistakes are almost distributed uniformly when the classes are non-hierarchical (e.g., CIFAR10, MNIST, STL10).

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

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

(4)

where $\tilde{Y}_2 = \{\tilde{y}_i\}_{i=1}^{m_2}$ and $\tilde{Y}_2 = \{\tilde{y}_i\}_{i=1}^{m_2}$ are uniformly selected to be sets of labels that disagree with $Y_2$ on $pm$ and $qm$ values (resp.). In addition, $\tilde{S}_2$ and $\tilde{S}_2$ are datasets obtained by replacing the labels of $S_2$ with $\tilde{Y}_2$ and $\tilde{Y}_2$ (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 $(1 - q)m$ 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. 2 and 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 $\delta^*_{\epsilon}$-uniform. Assume that $S_1, S_2 \sim P_{\mathcal{G}}(m)$. Let $h_{S_1}^\gamma$ be the output of the learning algorithm given access to a dataset $S_1$ and initialization $\gamma$. Then,

$$\mathbb{E}_{\tilde{Y}_2, \gamma}[\text{err}_{\gamma}(h_{S_1}^\gamma)] \leq \mathbb{P}_{S_1, S_2, \tilde{Y}_2} \left[ \mathbb{E}_{\gamma}[d^\ast_{\min}(h_{S_1}^\gamma)] \right. \geq d^\ast_{\min}(\mathcal{G}, S_1 \cup \tilde{S}_2) \left. + (1 + \alpha) p + \delta^1_{m} + \delta^2_{m,p,\alpha} \right),$$

(5)

where $\tilde{Y}_2 = \{\tilde{y}_i\}_{i=1}^{m_2}$ 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 $h^γ_{S_1}$, which is the term that we would like to minimize. The proposition assumes that the mistakes $h^γ_{S_2}$ generates on $X_2$ are distributed uniformly (with probability $\geq 1 - δ^m_{S_2}$). To account the likelihood that this assumption fails, our bound includes the term $δ^m_{S_2}$, which is assumed to be small. For empirical calculations of this bound, see the appendix.

Informally, to evaluate the performance of $h^γ_{S_2}$, the bound suggests the following idea. We start with an initial guess $p_m = p \in (0, 1/2)$ of the test error of $h^γ_{S_2}$. Using this guess, we compare its effective depth with the minimal NCC depth $d^m_{\min}(G, S_1 \cup S_2)$ required to NCC separate the samples in $S_1 \cup S_2$, where $S_2$ is the result of randomly relabeling $p_m m$ of $S_2$’s labels. Intuitively, if the mistakes of $h^γ_{S_1}$ are uniformly distributed and its effective depth $h^γ_{S_1}$ is smaller than $d^m_{\min}(G, S_1 \cup S_2)$, then, we expect $h^γ_{S_2}$ 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 minimal NCC depth required to NCC separate the samples in $S_1 \cup S_2$.

Next, we interpret each term separately. The term $E]\{d^m_{S_1}(h^γ_{S_1})\}$ depends on both the complexity of the classification problem and the implicit bias of SGD to favor networks of small effective depths. For example, if SGD does not minimize the effective depth or the labels in $S_1$ are statistically independent of the inputs for sufficiently large $m$, we expect $E]\{d^m_{S_1}(h^γ_{S_1})\} = L$. Simply put, $\hat{d}^m_{\min}(G, S_1 \cup S_2)$ measures the complexity of a task that involves fitting a dataset of size $2m$ samples, where $(2 - p_m)m$ of the labels are correct and $p_m m$ are random labels. By decreasing $p_m$, we expect $d^m_{\min}(G, S_1 \cup S_2)$ to decrease, making the first term in the bound larger. In addition, if $h = f^L \circ f$ is a neural network of a fixed width, $n_0$, it is impossible to fit an increasing amount of random labels without increasing $L$. Therefore, when $p_m m \to \infty$, the dataset $S_1 \cup S_2$ becomes increasingly hard to fit, and we expect $d^m_{\min}(G, S_1 \cup S_2)$ to tend to infinity. On the other hand, if $E]\{d^m_{S_1}(h^γ_{S_1})\}$ is bounded as a function of $L$ and $m$, when $p_m = 1/\sqrt{m}$, we obtain that $P]\{E]\{d^m_{S_1}(h^γ_{S_1})\} \geq d^m_{\min}(G, S_1 \cup S_2)\} \to 0$ and $p_m \to 0$, giving us $E]\{\text{err}_{\gamma}(h_{S_1})\} \leq δ^1_m + δ^2_m + o_m(1)$.

Interestingly, whenever our minimal depth hypothesis (Obs. [1]) holds, then $E]\{d^m_{S_1}(h^γ_{S_1})\}$ should be (relatively) unaffected by the depth $L$ of $h^γ_{S_1}$ as long as $L \geq L_0$. Therefore, in this regime, according to Prop. [1] the test performance of $h^γ_{S_1}$ should not decrease when increasing $L$ beyond $L_0$.

We note that the proposed generalization bound is fairly different from traditional generalization bounds [24, 21, 16]. 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 $\sqrt{m}$. These bounds assume that the complexity of the learned hypothesis is small in comparison to $\sqrt{m}$. We note that even in the presence of an implicit depth minimization, a standard parameter counting generalization bound would be vacuous. That is because, the overall number of parameters of the network after replacing the top, redundant, layers with a nearest class-center classifier would typically still exceed $\sqrt{m}$. On the other hand, Eq. [5] takes a different form. In this case, we do not require that the size of the network would be small in comparison with $\sqrt{m}$, rather the bound guarantees generalization if the effective size of the network is smaller than networks that fit partially random labels. Therefore, even if the effective size of the network is larger than $\sqrt{m}$, our bound may still provide a meaningful bound on the test error.

In general, computing the first term of the bound is generally impossible, due to the limited access to training data. However, instead, we can empirically estimate this term using a set of $k$ pairs $(S_{1i}^k, S_{2i}^k)$ of $m$ samples, yielding an additional term that scales as $O(1/\sqrt{k})$ to the bound (see Prop. [2] in the appendix).

4 Experiments

In this section, we experimentally analyze the emergence of neural collapse in the intermediate layers of neural networks. In the first experiment, we validate the “Minimal Depth Hypothesis” (Obs. [1]). In the second experiment, we consider the effect of corrupted labels on the extent of neural collapse, and specifically, on the effective depth. We show that the effective depth typically increases when extending the amount of corrupted labels present in the data. Throughout the experiments, we use
we also observe that the final NCC train/test accuracy rates of any intermediate layer converges as a function of $L$. The results of this experiment are extended and repeated with different architectures and datasets in the appendix.

Figure 1: Intermediate neural collapse of CONV-$L$-400 trained on CIFAR10. In the first and second rows we plot the CDNV and the NCC accuracy rates of neural networks with varying numbers of hidden layers evaluated on the train data (plotted in lin-log scale). Each line stands for a different layer within the network. In the third and fourth rows we report the same quantities over the test data.

Tesla-k80 GPUs for several hundred runs and each run took between 5-20 hours. For additional experiments, see the appendix.

4.1 Setup

Evaluation process. We consider $k$-class classification problems (e.g., CIFAR10) and train a multilayered neural networks $h = e \circ f^L = e \circ g^L \circ \cdots \circ g^1 : \mathbb{R}^n \rightarrow \mathbb{R}^C$ on some balanced training data $S$. The models are trained using cross-entropy loss minimization between its logits and the one-hot encodings of the labels. As a second stage, we evaluate the NCC train and test accuracy rates (i.e., $1 - \text{err}_S(h_i)$ and $1 - \text{err}_P(h_i)$) and the train and test class features variations ($\text{Avg}_{e \circ f \circ e}(V_{f_i}(S_c, S_c'))$ and $\text{Avg}_{e \circ f \circ e}(V_{f_i}(P_c, P_c'))$) for each sub-architecture $f^i = g^i \circ \cdots \circ g^1(x)$. The population distribution $P$ is replaced with the test set.

Hyperparameters. The optimizations were carried out using SGD with 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.

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

4.2 Results

Minimal depth in neural networks. To study the bias towards minimal depth, we trained a set of CONV-$L$-400 networks with varying depths. In Fig. 1 we report the CDNV on the train and test data and the NCC classification accuracy on the train and test data for networks of varying depths $L$. As can be seen, for networks with 8 or higher hidden layers, the eighth and higher layers have NCC train accuracy $\approx 100\%$, and therefore, are effectively of depth 7. In addition, we observe that neural collapse strengthens when increasing the network’s depth, on both train and test data. Furthermore, we also observe that the final NCC train/test accuracy rates of any intermediate layer converges when increasing $L$. We also observe that the NCC train/test accuracy rates of the intermediate layers converge as a function of $L$. The results of this experiment are extended and repeated with different architectures and datasets in the appendix.
Neural collapse 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 effective depths of models trained with varying amounts of corrupted labels. Namely, we study the degree of intermediate neural collapse in neural networks that are trained with varying amounts of corrupted labels. For concrete estimations of the bound, see AppendixA.

For this experiment we train instances of CONV-10-500 for CIFAR10 classification with 0%, 10%, 25%, 50% and 75% of the labels replaced with corrupted labels (e.g., uniformly distributed random labels). We compare the degrees of class features variability collapse and NCC separation in the intermediate layers of the trained networks and summarize the results in Fig.2. In the first and third rows we plot $\text{Avg}_{c \neq c'} [V_f(S_c, S_{c'})]$ and $\text{Avg}_{c \neq c'} [V_f(P_c, P_{c'})]$, for $i = 3, 4, 6, 8, 10$. In the second and fourth rows we plot the NCC accuracy rates $1 \text{ err}_S(h_i)$ and $1 \text{ err}_P(h_i)$.

As can be seen in Fig.2, consistent with the experiments in [15], we achieve a certain degree of neural collapse in the penultimate layer when training with or without corrupted labels. However, we observe several distinctions between the two cases. First, when training without corrupted labels, we obtain a higher degree of neural collapse across one of the layers. Furthermore, when training with 10% or 25% corrupted labels, the sixth layer’s NCC accuracy rate is lower than 98%, in contrast to training without 0% corrupted labels that gives us > 98% accuracy. Therefore, the $\epsilon$-empirical effective depth of the former network is 6 and the latter’s is 5, when $\epsilon = 0.02$ (see Def.1).

In Fig.3 we repeat the experiment with CONV-10-50 trained on MNIST with 0%, 1%, 2%, 5% and 10% of corrupted labels. We note that, as long as the amount of corrupted labels is < 10%, the models perfectly fit the training labels and achieve perfect NCC separability in their penultimate layers. Therefore, both properties are not indicative of the test performance. On the other hand, the two training regimes (with/without corrupted labels) are distinguishable when considering the degree of intermediate collapse on the train data. For example, the CDNV on the train data is significantly
lower across all layers when training without corrupted labels. In addition, the $\epsilon$-empirical effective depth of the network trained with 0% corrupted labels is 5 and for 2% it is 10 (for $\epsilon = 0.01$).

As a side note, we also notice (Figs. 2 and 3, fourth row) that the NCC classifiers corresponding to intermediate layers tend to be more resilient to corrupted labels than the model itself.

**Estimating the bound in Eq. 5.** We estimate the bound in Eq. 5 for CONV-10-50 and CONV-10-100 trained on MNIST and Fashion MNIST (resp.). For these estimation we used $\epsilon = 0.005$ by default. For each dataset we used different ‘guesses‘ $p$ (see Tab. 1) depending on the complexity of the learning task. We report an estimation of the expected test error of the models, $E_{S_{1, \gamma}}[\epsilon_{p}(h_{S_{1}})]$ and an estimation of the bound for each selection of $p$. As can be seen, for appropriate selections of $p$, we obtain non-trivial bounds on the test performance of the models, which are almost unheard of when it comes to standard bounds. Interestingly, if the guess $p$ is overoptimistic (e.g., close to $E_{S_{1, \gamma}}[\epsilon_{p}(h_{S_{1}})]$), then, the first term in the bound tends to be large compared to $E_{S_{1, \gamma}}[\epsilon_{p}(h_{S_{1}})]$.

For concrete technical details, see Appendix A.1

5 Conclusions

The role of depth and the ability to train neural networks that generalize well are not well understood. 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.
We study how the depth affects neural collapse. We characterize a notion of effective depth that measures the minimal layer that enjoys NCC separability. Furthermore, we hypothesize and empirically show that training favors neural networks of small effective depths.

As a next step, we study the connection between the effective NCC depth and generalization. We begin by addressing that neural collapse in the penultimate layer is not a good indication whether the network performs well on the test set. Then, we introduce a generalization bound that measures the likelihood in which the effective NCC depth of a trained neural network is (strictly) smaller than the minimal depth required to achieve NCC separability with partially corrupted labels. As we show empirically, this criterion is a good predictor of generalization.
Acknowledgements

This work was supported by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF – 1231216.

The authors would like to thank Tomaso Poggio, Andras György, Lior Wolf, Andrzej Banburski and X. Y. Han for illuminating discussions during the preparation of this manuscript.

References

[1] K. Arulkumaran, A. Cully, and J. Togelius. Alphastar: An evolutionary computation perspective, 2019. URL http://arxiv.org/abs/1902.01724 cite arxiv:1902.01724.

[2] 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.

[3] 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.

[4] 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. Chantzi, 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, M. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba. Evaluating large language models trained on code, 2021.

[5] 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.

[6] R. Eldan and O. Shamir. The power of depth for feedforward neural networks, 2016.

[7] T. Ergen and M. Pilanci. Revealing the structure of deep neural networks via convex duality. In M. Meila and T. Zhang, editors, Proceedings of the 38th International Conference on Machine Learning, volume 139 of Proceedings of Machine Learning Research, pages 3004–3014. PMLR, 18–24 Jul 2021.

[8] C. Fang, H. He, Q. Long, and W. 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.

[9] 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.

[10] X. Han, V. Papan, and D. L. Donoho. Neural collapse under MSE loss: Proximity to and dynamics on the central path. In International Conference on Learning Representations, 2022. URL https://openreview.net/forum?id=w1UbdvWHR3.

[11] 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.

[12] K. He et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet. In ICCV, 2015.

[13] L. Hui, M. Belkin, and P. Nakrirkian. Limitations of neural collapse for understanding generalization in deep learning. arXiv preprint arXiv:2202.08584, 2022.
[14] J. Lu and S. Steinerberger. Neural collapse with cross-entropy loss. *CoRR*, abs/2012.08465, 2020.
[15] D. G. Mixon, H. Parshall, and J. Pi. Neural collapse with unconstrained features, 2020.
[16] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012. ISBN 026201825X.
[17] V. Papyan, 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.
[18] T. Poggio, A. Banburski, and Q. Liao. Theoretical issues in deep networks. *Proceedings of the National Academy of Sciences*, 2020.
[19] A. Rangamani, M. Xu, A. Banburski, Q. Liao, T. Galanti, and T. Poggio. Dynamics and neural collapse in deep classifiers trained with the square loss. Technical report, Center for Brains, Minds and Machines (CBMM), 2022.
[20] I. Safran, R. Eldan, and O. Shamir. Depth separations in neural networks: What is actually being separated?, 2021.
[21] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press, 2014. ISBN 978-1-10-705713-5.
[22] 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.
[23] 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.
[24] V. N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.
[25] 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.
[26] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer. Scaling vision transformers, 2021.
[27] Z. Zhu, T. DING, J. Zhou, X. Li, C. You, J. Sulam, and Q. Qu. A geometric analysis of neural collapse with unconstrained features. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=KRODJAa6pzE.
A Additional Experiments

Auxiliary experiments on the effective depth. We repeated the experiment in Fig. 1 in the main text. In Figs. 4 and 5 we plot the results of the same experiment, with different networks and datasets (see captions). 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, similar to the results in Fig. 1 in the main text, the degree of neural collapse improves with the network’s depth.

Auxiliary experiments with noisy labels. We repeated the experiment in Figs. 2 and 3 in the main text. In Figs. 6 and 7 we 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.

The effect of the width on intermediate collapse. As an additional experiment, we studied the effect of the width on intermediate neural collapse. In Fig. 10 we plot the results of this experiment, for CONV-10-H networks, with $H = 20, 40, 80, 160, 320$. 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 a different width. As can be seen, intermediate neural collapse strengthens when increasing the width of the neural network, on both train and test data.

A.1 Estimating the Generalization Bound

In Prop. 1 we introduce a generalization bound for deep neural networks. In this section we empirically estimate the bound and demonstrate non-trivial estimations of the test performance.

The results are summarized in Tab. 2. We report (an estimation of) the mean test error, the choices of $\epsilon$ and $p$, the estimations of the first term in Eq. 5 and the full bound in Eq. 5. We experiment with multiple values of $p$ depending on the hardness of the given task. We chose $\epsilon = 0.005$ as our default threshold for deciding whether we have separation or not.

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

$$
\mathbb{P}_{S_1, S_2, Y_2} \left[ \mathbb{E}_y \left[ \delta_{S_1}^e \left( h_{S_1}^\gamma \right) \right] \geq \delta_{\min}^e \left( G, S_1 \cup \tilde{S}_2 \right) \right].
$$

(6)

According to Prop. 2, in order to estimate this term we need to generate i.i.d. triplets $(S_1, S_2, Y_2)$. Since we have a limited access to training data, we use a variation of cross-validation and generate $k_1 = 5$ i.i.d. disjoint splits $(S_1^1, S_2^1)$ of the training data $S$. For each one of these pairs, we generate $k_2 = 3$ corrupted labelings $Y_2^1$. We denote by $S_2^1$ the set obtained by replacing the labels of $S_2^1$ with $Y_2^1$ and $\tilde{S}_2^1 := S_1^1 \cup \tilde{S}_2^1$.

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

As a next step, we would like to evaluate $\mathbb{I}[d_i \geq \delta_{\min}^e \left( G, S_1^1 \cup \tilde{S}_2 \right)]$. We notice that $d_i \geq \delta_{\min}^e \left( G, S_1^1 \cup \tilde{S}_2 \right)$ if and only if there is a $d_i$-layered neural network $f = g^{d_i} \circ \cdots \circ g^1$ for which $\text{err}_{S_1^1} \left( h \right) \leq \epsilon$, where $h(x) := \text{arg min}_{c \in [C]} \| f(x) - \mu_f (S_c) \|$. 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 $\epsilon$-NCC separable, i.e., $\text{err}_{S_1^1} \left( h \right) \leq \epsilon$, where $h(x) := \text{arg min}_{c \in [C]} \| f(x) - \mu_f (S_c) \|$. If SGD implicitly optimizes neural networks to maximize NCC separability as observed in [17] (and also in this paper), we should expect to obtain $\epsilon$-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 $\delta_{S_1^1}^e \left( h_{S_1^1}^\gamma \right)$ with $\min_{t \in [T_2]} \delta_{S_1^1}^e \left( h_t \right)$ and estimate $\mathbb{I}[d_i \geq \delta_{\min}^e \left( G, S_1^1 \cup \tilde{S}_2 \right)]$ using $\mathbb{I}[d_i \geq \min_{t \in [T_2]} \delta_{S_1^1}^e \left( h_t \right)]$. 

13
Our final estimation is the following

\[ \frac{1}{k_1} \sum_{i=1}^{k_1} \frac{1}{k_2} \sum_{j=1}^{k_2} \mathbb{1} \left[ d_i \geq \min_{t \in [T_2]} d_{S_i}^\gamma (h_t) \right] \approx \mathbb{P}_{S_1, S_2, \bar{S}_2} \left[ \mathbb{E}_{\gamma} \left[ d_{S_i}^\gamma (h_{\gamma}^S) \right] \geq d_{\min}^\gamma (G, S_1 \cup \bar{S}_2) \right]. \]  

(7)

In order to estimate the bound we assume that \( \delta_m^1 \) and \( \delta_m^2 \) are negligible constants and that \( \alpha = 1 \). The estimation of the bound is given by the sum of the LHS in Eq. 7 and \( p \).

**Estimating the mean test error.** To estimate the mean test error, \( \mathbb{E}_{S_1, \gamma} [\text{err}_P(h_{\gamma}^S)] \), 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 \( \gamma \) with averages across the \( k_1 = 5 \) random selections of \( \{S_i\}_{i=1}^{k_1} \) and \( T_1 = 5 \) random selections of \( \{\gamma_t\}_{t=1}^{T_1} \). Namely, we compute the following

\[ \frac{1}{k_1} \sum_{i=1}^{k_1} \frac{1}{T_1} \sum_{t=1}^{T_1} \text{err}_{S_{test}}(h_{\gamma_t}^S) \approx \mathbb{E}_{S_1, \gamma} [\text{err}_P(h_{\gamma}^S)]. \]
Figure 4: Intermediate neural collapse of CONV-L-400 trained on CIFAR10. See Fig. [in the main text for details.]
Figure 5: Intermediate neural collapse of MLP-$L$-$300$ trained on CIFAR10. See Fig. in the main text for details.
Figure 6: **Intermediate neural collapse of CONV-L-50 trained on MNIST.** See Fig. [1] in the main text for details.
Figure 7: Intermediate neural collapse of MLP-\(L-100\) trained on Fashion MNIST. See Fig. [1] in the main text for details.
Figure 8: **Intermediate neural collapse of MLP-10-500 trained on CIFAR10 with noisy labels.**
See Fig. 2 in the main text for details.

Figure 9: **Intermediate neural collapse of CONV-10-100 trained on Fashion MNIST with noisy labels.**
See Fig. 2 in the main text for details.
Figure 10: **Intermediate neural collapse of CONV-10-$H$ trained on MNIST when varying the width.** See Fig. [1](#) in the main text 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 $\delta_m^1$-uniform. Assume that $S_1, S_2 \sim P_B(m)$. Let $h_{S_1}^\gamma$ be the output of the learning algorithm given access to a dataset $S_1$ and initialization $\gamma$. Then,

$$
E_{S_1, \gamma}[err_P(h_{S_1}^\gamma)] \leq \mathbb{P}_{S_1, S_2, \gamma_2} \left[ E_{\gamma_1}[d_{S_1}(h_{S_1}^\gamma)] \geq d_{min}(G, S_1 \cup \tilde{S}_2) \right] + (1 + \alpha) p + \delta_m^1 + \delta_m^{2, p, \alpha},
$$

(5)

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.

**Proof.** Let $S_1 = \{(x_1, y_1)\}_{i=1}^m$ and $S_2 = \{(x_2, y_2)\}_{i=1}^m$ be two balanced datasets. Let $\epsilon > 0$, $p > 0$ and $q \geq (1 + \alpha) p$. Let $Y_2$ and $\tilde{Y}_2$ 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,

$$
A_1 = \{(S_1, S_2, Y_2) \mid \exists q \geq (1 + \alpha) p : d_{min}(G, S_1 \cup \tilde{S}_2) > E_{\gamma_2}[d_{min}(G, S_1 \cup \hat{S}_2)]\}
$$

$$
A_2 = \{(S_1, S_2) \mid \text{the mistakes of } h_{S_1}^\gamma \text{ are not uniform over } S_2\}
$$

$$
A_3 = \{(S_1, S_2, \tilde{Y}_2) \mid (S_1, S_2, \tilde{Y}_2) \notin A_1 \cup A_2 \text{ and } E_{\gamma}[d_{\hat{S}_1}^\epsilon(h_{S_1}^\gamma)] < d_{min}(G, S_1 \cup \tilde{S}_2)\}
$$

$$
A_4 = \{(S_1, S_2, \hat{Y}_2) \mid (S_1, S_2, \hat{Y}_2) \notin A_1 \cup A_2 \text{ and } E_{\gamma}[d_{\tilde{S}_1}^\epsilon(h_{S_1}^\gamma)] \geq d_{min}(G, S_1 \cup \hat{S}_2)\}
$$

$$
B_1 = \{(S_1, S_2, Y_2) \mid E_{\gamma}[d_{\tilde{S}_1}^\epsilon(h_{S_1}^\gamma)] \geq d_{min}(G, S_1 \cup \tilde{S}_2)\}
$$

By the law of total expectation

$$
E_{S_1, \gamma}[err_P(h_{S_1}^\gamma)] = E_{S_1, S_2, \gamma_2}[E_{\gamma}[err_{\hat{S}_2}(h_{S_1}^\gamma)]]
$$

$$
= \sum_{i=1}^4 \mathbb{P}[A_i] \cdot E_{S_1, S_2, \gamma_2}[E_{\gamma}[err_{\hat{S}_2}(h_{S_1}^\gamma)] \mid A_i]
$$

$$
\leq \mathbb{P}[A_1] + \mathbb{P}[A_2] + E_{S_1, S_2, \gamma_2}[err_{\hat{S}_2}(h_{S_1}^\gamma)] \mid \mathbb{P}[A_3] + \mathbb{P}[B_1],
$$

where the last inequality follows from $err_{\hat{S}_2}(h_{S_1}^\gamma) \leq 1$, $\mathbb{P}[A_3] \leq 1$ 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 $\delta_m^1$-uniform, $\mathbb{P}[A_2] \leq \delta_m^1$. In addition, by definition $\mathbb{P}[A_1] \leq \delta_m^{2, p, \alpha}$.

As a next step, we upper bound $E_{S_1, S_2, \gamma_2}[\mathbb{P}[A_3] \mid A_3]$. Assume that $(S_1, S_2, \tilde{Y}_2) \in A_3$. Hence, $(S_1, S_2, \hat{Y}_2) \notin A_1 \cup A_2$. Then, the mistakes of $h_{\hat{S}_1}^\gamma$ over $S_2$ are uniformly distributed (with respect to the selection of $\gamma$). Assume by contradiction that $q_{m, \gamma} := err_{\hat{S}_2}(h_{\hat{S}_1}^\gamma) > (1 + \alpha) p$ for some initialization $\gamma$. Then, since the mistakes of $h_{\hat{S}_1}^\gamma$ over $S_2$ are uniformly distributed, $q_m = err_{\hat{S}_2}(h_{\hat{S}_1}^\gamma) > (1 + \alpha) p$ for all initializations $\gamma$. Therefore, we have

$$
E_{\tilde{Y}_2}[d_{min}(\mathcal{F}, S_1 \cup \tilde{S}_2)] \leq E_{\gamma}[d_{\tilde{S}_1}^\epsilon(h_{S_1}^\gamma)] < d_{min}(G, S_1 \cup \tilde{S}_2),
$$

where the first inequality follows from the definition of $d_{min}(\mathcal{F}, S_1 \cup \tilde{S}_2)$ 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 $(\tilde{Y}_2, \hat{Y}_2) \notin A_1$. Therefore, we conclude that in this case, $q = E_{\gamma}[err_{\tilde{S}_2}(h_{S_1}^\gamma)] \leq (1 + \alpha) p$ and $E_{S_1, S_2, \gamma_2}[err_{\hat{S}_2}(h_{S_1}^\gamma)] \mid A_3 \leq (1 + \alpha) p$.

□

**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 $\delta_m^1$-uniform. Let $S_1, S_2, S_1', S_2' \sim P_B(m)$ (for $i \in [k]$). Let $Y_2 = \{\tilde{y}_i\}_{i=1}^m$ be a set of labels that disagrees with $Y_2$ on uniformly selected $pm$ labels and $S_2^i$ is a relabeling of $S_2$ with the labels in $Y_2^i$. Let $h_{S_1}^\gamma$ be the output of the learning algorithm given access to a dataset $S_1$ and initialization $\gamma$. Then, with probability at least $1 - \delta$ over the selection of $\{(S_1, S_2, Y_2^i)\}_{i=1}^k$,

$$
E_{S_1, \gamma}[err_P(h_{S_1}^\gamma)] \leq \frac{1}{k} \sum_{i=1}^k \left[ E_{\gamma}[d_{\tilde{S}_1}^\epsilon(h_{S_1}^\gamma)] \geq d_{min}(G, S_1' \cup S_2^i) \right]
$$

$$
+ (1 + \alpha) p + \delta_m^1 + \delta_m^{2, p, \alpha} + \sqrt{\frac{\log(2/\delta)}{2k}}.
$$
Proof. By Prop. 1, we have

\[ \mathbb{E}_{S_1} \mathbb{E}_{\gamma} [\text{err}_P(h_{S_1}^\gamma)] \leq \mathbb{P}_{S_1, S_2, \tilde{Y}_2} \left[ \mathbb{E}_{\gamma} [d'_{S_1} (h_{S_1}^\gamma)] \geq d'_{\text{min}} (G, S_1 \cup \tilde{S}_2) \right] + (1 + \alpha) p_m + \delta^1_m + \delta^2_{m, p, \alpha} \]

We define i.i.d. random variables

\[ V_i = \mathbb{I} \left[ \mathbb{E}_{\gamma} [d'_{S_1} (h_{S_1}^\gamma)] \geq d'_{\text{min}} (G, S_1 \cup \tilde{S}_2) \right]. \quad (10) \]

Therefore, we can rewrite,

\[ \mathbb{P}_{S_1, S_2, \tilde{Y}_2} \left[ \mathbb{E}_{\gamma} [d'_{S_1} (h_{S_1}^\gamma)] \geq d'_{\text{min}} (G, S_1 \cup \tilde{S}_2) \right] = \mathbb{E} [V_1]. \quad (11) \]

By Hoeffding’s inequality,

\[ \Pr \left[ k^{-1} \sum_{i=1}^{k} V_i - \mathbb{E}[V_1] \geq \epsilon \right] \leq 2 \exp \left( -2k\epsilon^2 \right). \quad (12) \]

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

\[ \mathbb{E}[V_1] \leq \frac{1}{k} \sum_{i=1}^{k} V_i + \sqrt{\log(1/2\delta) / 2k}. \quad (13) \]

When combined with Prop. 1, we obtain the desired bound. \( \square \)