CENTER FOR
Brains
Minds+
Machines

CBMM Memo No. 117 **October 9, 2022**

Deep Classifiers trained with the Square Loss

Mengjia Xu[1,2], **Akshay Rangamani**[1], **Andrzej Banburski**[1], **Qianli Liao**[1], **Tomer Galanti**[1], **Tomaso Poggio**[1]
[1]

[1]Center for Brains, Minds and Machines, MIT,
[2]Division of Applied Mathematics, Brown University

Abstract

We overview several properties – old and new – of training overparametrized deep homogeneous RELU networks under the square loss. We study the convergence to a solution with minimum ρ, which is the product of the Frobenius norms of each layer weight matrix, when normalization by a Lagrange multiplier (LM) is used together with Weight Decay (WD) under forms of gradient descent. In the absence of LM + WD, good solutions for classification may still be achieved because of the implicit bias towards small norm solutions in the gradient dynamics introduced by close-to-zero initial conditions on the norms of the weights. Classical bounds show that for a specific network, the quasi-interpolating solutions with smaller ρ have better margin and better bounds on the expected classification error. Furthermore, we derive theoretically novel norm-based bounds for convolutional layers: despite overparametrization, they turn out to be still loose but almost non-vacuous, at least in some of our experiments. This result is new, as far as we know, and is especially interesting because similar bounds for dense networks are orders of magnitude worse. Next we prove that quasi-interpolating solutions obtained by SGD in the presence of WD have a bias towards low rank weight matrices. The same analysis predicts the existence of an inherent SGD noise for deep networks under the same conditions. In both cases, we verify our predictions experimentally. We also prove that our model have the recently discovered behavior of Neural Collapse and predict its properties without any specific assumption – unlike other published proofs. Our analysis supports the idea that the advantage of deep networks relative to other standard classifiers is greater for the problems to which sparse deep architectures such as CNNs can be applied. The deep reason is that CNNs reflect the function graph of certain compositionally sparse target functions and thus can be approximated well by "sparse" networks without incurring in the curse of dimensionality.

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

Dynamics in Deep Classifiers Trained with the Square Loss: Normalization, Low Rank, Neural Collapse and Generalization Bounds

Mengjia Xu[1,2], Akshay Rangamani[1], Andrzej Banburski[1], Qianli Liao[1], Tomer Galanti[1], Tomaso Poggio[1]

[1]Center for Brains, Minds and Machines, MIT
[2]Division of Applied Mathematics, Brown University

December 7, 2022

Abstract

We overview several properties – old and new – of training overparametrized deep networks under the square loss. We first consider a model of the dynamics of gradient flow under the square loss in deep homogeneous ReLU networks. We study the convergence to a solution with the absolute minimum ρ, which is the product of the Frobenius norms of each layer weight matrix, when normalization by Lagrange multipliers (LM) is used together with Weight Decay (WD) under different forms of gradient descent. A main property of the minimizers that bounds their expected error *for a specific network architecture* is ρ. In particular, we derive novel norm-based bounds for convolutional layers that are orders of magnitude better than classical bounds for dense networks. Next we prove that quasi-interpolating solutions obtained by Stochastic Gradient Descent (SGD) in the presence of WD have a bias towards low rank weight matrices – that, as we also explain, should improve generalization. The same analysis predicts the existence of an inherent SGD noise for deep networks. In both cases, we verify our predictions experimentally. We then predict Neural Collapse and its properties without any specific assumption – unlike other published proofs. Our analysis supports the idea that the advantage of deep networks relative to other classifiers is greater for the problems that are appropriate for sparse deep architectures such as CNNs. The deep reason compositionally sparse target functions can be approximated well by "sparse" deep networks without incurring in the curse of dimensionality.

1 Introduction

A widely held belief in the last few years has been that the cross-entropy loss is superior to the square loss when training deep networks for classification problems. As such, the attempts at understanding the theory of deep learning has been largely focused on exponential-type losses (1; 2), like the cross-entropy. For these losses, the predictive ability of deep networks depends on the implicit complexity control of Gradient Descent algorithms that leads to asymptotic maximization of the classification margin on the training set (3; 1; 4). Recently however, (5) has empirically demonstrated that it is possible to achieve a similar level of performance, if not better, using the square loss, paralleling older results for Support Vector Machines (SVMs) (6). Can a theoretical analysis explain when and why regression should work well for classification? This question was the original motivation for this paper and preliminary versions of it (7; 8).

In deep learning binary classification, unlike the case of linear networks, we expect from previous results (in the absence of regularization) several global minima with zero square loss, thus corresponding to interpolating solutions (in general degenerate, see (9; 10) and reference therein), because of overparametrization. Although all the interpolating solutions are optimal solutions to the regression problem, they will in general correspond to different (normalized) margins and to different expected classification performance. In other words, zero square loss does not imply by itself neither large

margin nor good classification on a test set. When can we expect the solutions to the regression problem obtained by Gradient Descent (GD) to have a large margin?
We introduce a simplified model of the training procedure that uses square loss, binary classification, gradient flow and Lagrange multipliers (LM) for normalizing the weights. With this model we show that obtaining large margin interpolating solutions depends on the scale of initialization of the weights close to zero, in the absence of regularization (also called weight decay). Assuming convergence, we describe the qualitative dynamics of the deep network's parameters and show that ρ, which is the product of the Frobenius norms of the weight matrices, grows non-monotonically until a large margin, that is small ρ solution is found reached. Assuming that local minima and saddle points can be avoided, this analysis suggests that with weight decay (or sometimes with just small initialization), gradient descent techniques may yield convergence to a minimum with a ρ biased to be small.
In the presence of weight decay, perfect interpolation of all data points cannot occur and is replaced by quasi-interpolation of the labels. In the special case of binary classification case in which $y_n = \pm 1$, quasi-interpolation is defined as $\forall\, n :\ |f(x_n) - y_n| \leq \epsilon$, where $\epsilon > 0$ is small. Our experiments and analysis of the dynamics show that, in the presence of regularization, there is a weaker dependence on initial conditions, as has been observed in (5). We show that weight decay helps stabilize normalization of the weights, in addition to its role in the dynamics of the norm.
We then apply basic bounds on expected error to the solutions provided by SGD (for weight decay $\lambda > 0$), which have locally minimum ρ. For normal training set sizes, the bounds are still vacuous but much closer[1] to the test error than previous estimates. This is encouraging because in our setup large overparametrization, corresponding to interpolation of the training data (11), coexists with a relatively small Rademacher complexity *because of the sparsity induced by the locality of the convolutional kernel*.
We then turn to show that the quasi-interpolating solutions satisfy the recently discovered Neural Collapse (NC) phenomenon (12), assuming SGD with minibatches. According to Neural Collapse, a dramatic simplification of deep network dynamics takes place – not only do all the margins become very similar to each other, but the last layer classifiers and the penultimate layer features form the geometrical structure of a simplex equiangular tight frame (ETF). Here we prove the emergence of Neural Collapse for the square loss for the networks we study — without any additional assumption (such as *unconstrained features*).
Finally, the study of SGD reveals surprising differences relative to GD. In particular, in the presence of regularization, SGD does not converge to a perfect equilibrium: there is always, at least generically, SGD noise.The underlying reason is a rank constraint that depends on the size of the minibatches. This also implies an SGD bias towards small rank solutions that reinforces a similar bias due to maximization of the margin under normalization (that can be inferred from (13)).

Contributions The main original contributions in this paper are

- We analyze the dynamics of deep network parameters, their norm, and the margins under gradient flow on the square loss, using *Lagrange normalization* (*LN*). We describe the evolution of ρ, and the role of Weight Decay and normalization in the training dynamics. The analysis in terms of Lagrange multipliers of the dynamics in the "polar" coordinates ρ, V_k is new. Many of the observed properties are not. Arguably, our analysis of the bias towards minimum ρ and its dynamics with and without weight decay is an original contribution.

- Our norm-based generalization bounds for CNNs are new. We outline in this paper a derivation for the case of non-overlapping convolutional patches. The extension to the general case follows naturally and will be described in a forthcoming paper. The bounds show that generalization for CNNs can be orders of magnitude better than for dense networks. In the experiments we describe, the bounds turn out to be loose but close to non-vacuous. They appear to be much better than the other empirical tests of generalization bounds – all for dense networks – that we know of. The main reason for this, in addition to the relatively simple task (binary classification in CIFAR) is the sparsity of the convolutional network, that is the low dimensionality (or locality) of the kernel.

- We prove that convergence of gradient descent optimization with weight decay and normalization yields Neural Collapse for deep networks trained with square loss in the binary as well as in

[1] by several orders of magnitude!

the multiclass classification case. Experiments verify the predictions. Our proof is free of any assumption – unlike other recent papers that depend on the "unconstrained feature assumption".

- We prove that training the network using SGD with weight decay induces a bias towards low-rank weight matrices. As we will describe in a separate paper low rank can yield better generalization bounds.
- The same theoretical observation that predicts a low-rank-bias also predicts the existence of an intrinsic SGD noise in the weight matrices and in the margins.

2 Related Work

There has been much recent work on the analysis of deep networks and linear models trained using exponential-type losses for classification. The implicit bias of Gradient Descent towards margin maximizing solutions under exponential type losses was shown for linear models with separable data in (14) and for deep networks in (1; 2; 15; 16). Recent interest in using the square loss for classification has been spurred by the experiments in (5), though the practice of using the square loss is much older (6). Muthukumar et. al. (17) recently showed for linear models that interpolating solutions for the square loss are equivalent to the solutions to the hard margin SVM problem (see also (7)). Recent work also studied interpolating kernel machines (18; 19) which use the square loss for classification.
In the recent past, there have been a number of papers analyzing deep networks trained with the square loss. These include (20; 21) that show how to recover the parameters of a neural network by training on data sampled from it. The square loss has also been used in analyzing convergence of training in the Neural Tangent Kernel (NTK) regime (22; 23; 24). Detailed analyses of two-layer neural networks such as (25; 26; 27) typically use the square loss as an objective function. However these papers do not specifically consider the task of classification.
A large effort has been spent in understanding generalization in deep networks. The main focus has been solving the puzzle of how overparametrized deep networks (with more parameters than data) are able to generalize. An influential paper (11) showed that overparametrized deep network that usually fit randomly labeled data also generalize well when they trained on correctly labeled data. Thus the training error does not give any information about test error: there is no *uniform convergence* of training error to test error. This is related to another property of overparametrization: standard VC bounds are always vacuous when the number of parameters is larger than the number of data. Though often forgotten, it is however well known that another type of bounds – on the norm of parameters– may provide generalization even if there are more parameters than data. This point was made convincingly in (28) which provides norm-based bounds for deep networks[2]. Bartlett bounds and related ones (29; 30) in practice turn out to be very loose. Empirical studies such as (31) found little evidence so far that norms and margins correlate well with generalization.
Neural Collapse (NC) (12) is a recently discovered empirical phenomenon that occurs when training deep classifiers using the cross-entropy loss. Since its discovery, there have been a few papers analytically proving its emergence when training deep networks. Mixon et. al. (32) show NC in the regime of "unconstrained features". Recent results in (33) perform a more comprehensive analysis of NC in the unconstrained features paradigm. There have been a series of papers analytically showing the emergence of NC when using the cross-entropy loss (34; 35; 36). In the study of the emergence of NC when training using the square loss, Ergen and Pilanci (37) (see also (38)) derived it through a convex dual formulation of deep networks. In addition to that, (39) and (40) show the emergence of NC in the unconstrained features regime. Our independent derivation is different from these approaches, and shows that NC emerges in the presence of normalization and weight decay.
Several papers in recent years have studied the relationship between implicit regularization in linear neural networks and rank minimization. A main focus was on the matrix factorization problem, which corresponds to training a depth-2 linear neural network with multiple outputs w.r.t. the square loss (see references in (13)). Beyond factorization problems, it was shown that in linear networks of output dimension 1, gradient flow w.r.t. exponential-type loss functions converges to networks where the weight matrix of every layer is of rank 1. However, for nonlinear neural networks things are less clear. Empirically, several studies (see references in (13)) showed that replacing the weight matrices

[2]The focus of this paper on ρ is directly related

by low-rank approximations results in only a small drop in accuracy. This suggests that the weight matrices in practice are not too far from being low-rank.

3 Problem Setup

In this section, we describe the training settings considered in our work. We study training deep neural network with ReLU non-linearity using square loss minimization for classification problems. In the proposed analysis, we apply a specific normalization technique: Weight Normalization, which is equivalent to Lagrange multiplier, as well as regularization (also called Weight Decay), since such mechanisms seem commonly used for reliably training deep networks using gradient descent techniques (41; 5).

3.1 Assumptions

Throughout the theoretical analysis we make in some places simplifying assumptions relative to standard practice in deep neural networks. We mostly consider the case of binary classification though our analysis of Neural Collapse includes multiclass classification. We restrict ourselves to the square loss. We consider gradient descent techniques but we assume different forms of them in various sections of the paper. In the first part, we assume continuous Gradient Flow (GF) instead of GD or SGD. Gradient flow is the limit of discrete Gradient Descent algorithm with the learning rate being infinitesimally small (we describe an approximation of Gradient Descent within a Gradient Flow approach in (8)). SGD is specifically considered and shown to bias rank and induce asymptotic noise that is unique to it. The analysis of Neural Collapse is carried out using SGD with small learning rates. Furthermore, we assume weight normalization by a Lagrange multiplier term added to the loss function, that normalizes the weight matrices. This is equivalent to Weight Normalization but is not equivalent to the more commonly used Batch Normalization.
We also assume throughout that the network is overparametrized and so that there is convergence to global minima with appropriate initialization, parameter values and data.

3.2 Classification with Square Loss Minimization

In this work we consider a square loss minimization for classification along with regularization and weight normalization. We consider a binary classification problem given a training dataset $\mathcal{S} = \{(x_n, y_n)\}_{n=1}^N$, where $x_n \in \mathbb{R}^d$ are the inputs (normalized such that $\|x_n\| \leq 1$) and $y_n \in \{\pm 1\}$ are the labels. We use deep rectified homogeneous networks with L layers to solve this problem. For simplicity, we consider networks $f_W : \mathbb{R}^d \to \mathbb{R}^p$ of the following form $f_W(x) = W_L\sigma\left(W_{L-1}\ldots\sigma\left(W_1x\right)\ldots\right)$, where $x \in \mathbb{R}^d$ is the input to the network and $\sigma : \mathbb{R} \to \mathbb{R}$, $\sigma(x) = \max(0, x)$ is the rectified linear unit (ReLU) activation function that is applied coordinate-wise at each layer. The last layer of the network is linear (see Figure 1).
Due to the positive homogeneity of ReLU (i.e., $\sigma(\alpha x) = \alpha\sigma(x)$ for all $x \in \mathbb{R}$ and $\alpha > 0$), one can reparametrize $f_W(x)$ by considering normalized[3] weight matrices $V_k = \frac{W_k}{\|W_k\|}$ and define $\rho_k = \|W_k\|$ obtaining $f_W(x) = \rho_L V_L\sigma\left(\rho_{L-1}\ldots\sigma\left(\rho_1V_1x\right)\ldots\right)$. Because of homogeneity of the ReLU it is possible to pull out the product of the layer norms as $\rho = \prod_k \rho_k$ and write $f_W(x) = \rho f_V(x) = \rho V_L\sigma\left(V_{L-1}\ldots\sigma\left(V_1x\right)\ldots\right)$. Notice that the two networks – $f_W(x)$ and $\rho f_V(x)$ – are equivalent reparameterizations of the same function (if $\rho = \prod_k \rho_k$) but their optimization generally differ. We define $f_n := f_V(x_n)$.
We adopt in our definition the convention that the norm ρ_j of the convolutional layers is *the norm of their filters* and not the norm of their associated Toeplitz matrices. The reason is that this what our novel bounds for CNNs state (see also section 3.3 in (42) and (43)). The total ρ calculated in this way is the quantity that enters the generalization bounds of section 4.
In practice, certain normalization techniques are used in order to train neural networks. This is usually performed using either batch normalization (BN) or, less frequently, weight normalization (WN). BN consists of standardizing the output of the units in each layer to have zero mean and unit variance wrt training set. WN normalizes the weight matrices (section 10 in (4)). In our analysis, we model normalization by normalizing the weight matrices, using a *Lagrange multiplier* (*LM*) term added to the loss function. This approach is equivalent to WN.

[3]We choose the Frobenius norm here.

In the presence of normalization, we assume that all layers are normalized, except for the last one, via the added Lagrange multiplier. Thus, the weight matrices $\{V_k\}_{k=1}^L$ are constrained by the Lagrange multiplier term to be close to, and eventually converge to, unit norm matrices (in fact to fixed norm matrices); notice that normalizing V_L and then multiplying the output by ρ, is equivalent to letting $W_L = \rho V_L$ be unnormalized. Thus, f_V is the network that at convergence has $L-1$ normalized layers (see Figure 1).
We can write the Lagrangian corresponding to the minimization of the regularized loss function under the constraint $\|V_k\|^2 = 1$ in the following manner

$$\begin{aligned}\mathcal{L}_{\mathcal{S}}(\rho, \{V_k\}_{k=1}^L) :&= \frac{1}{N}\sum_n (\rho f_n - y_n)^2 + \sum_{k=1}^L \nu_k(\|V_k\|^2 - 1) + \lambda\rho^2 \\ &= \frac{1}{N}\sum_n (1 - \rho\bar{f}_n)^2 + \sum_{k=1}^L \nu_k(\|V_k\|^2 - 1) + \lambda\rho^2,\end{aligned} \tag{1}$$

where ν_k are the Lagrange multipliers and $\lambda > 0$ is a predefined parameter.

Separability and Margins. Two important aspects of classification are *separability* and *margins*. For a given sample (x, y) (train or test sample) and model f_W, we say that f_W correctly classifies x, if $\bar{f}_n = y_n f_n > 0$. In addition, for a given dataset $\mathcal{S} = \{(x_n, y_n)\}_{n=1}^N$, *separability* is defined as the condition in which all training samples are classified correctly, $\forall\, n \in [N] :\ \bar{f}_n > 0$. Furthermore, when $\sum_{n=1}^N \bar{f}_n > 0$, we say that *average separability* is satisfied. The minimum of $\mathcal{L}_{\mathcal{S}}$ for $\lambda = 0$ is usually zero under our assumption of overparametrization. This corresponds to separability.
Notice that if f_W is a zero loss solution of the regression problem, then $\forall\, n :\ f_W(x_n) = y_n$, which is also equivalent to $\rho f_n = y_n$, where we call $y_n f_n = \bar{f}_n$ *the margin*[4] *for* x_n. By multiplying both sides of this equation by y_n, and summing both sides over $n \in [N]$, we obtain that $\rho \sum_n \bar{f}_n = N$. Thus, the norm ρ of a minimizer is inversely proportional to its average margin μ in the limit of $\lambda = 0$, with $\mu = \frac{1}{N}\sum_n \bar{f}_n$. It is also useful to define *the margin variance* $\sigma^2 = M - \mu^2$ *with* $M = \frac{1}{N}\sum_n \bar{f}_n^2$. Notice that $M = \frac{1}{N}\sum_n \bar{f}_n^2 = \sigma^2 + \mu^2$ and that both M and σ^2 are not negative.

Interpolation and Quasi-interpolation. Assume that the weights V_k are normalized at convergence. Then

Lemma 1 *For $\lambda = 0$ there are solutions that interpolate all data points with the same margin and achieve zero loss. For $\lambda > 0$ there are no solutions that have the same margins and interpolate. However there are solutions with the same margins that quasi-interpolate and are critical points of the gradient.*

Proof Consider the loss $\mathcal{L}_{\mathcal{S}} = \frac{1}{N}\sum_n (1-\rho\bar{f}_n)^2 + \lambda\rho^2 = 1 - 2\rho\mu + \rho^2 M + \lambda\rho^2$. For $\lambda = 0$, a zero of the loss $\mathcal{L}_{\mathcal{S}} = 0$ implies $\forall\, n \in [N] :\ \mu = \bar{f}_n$ and $\mu = \frac{1}{\rho}$. However, for $\lambda > 0$ the assumption that all $\bar{f}_n$ are equal yields $M = \mu^2$ and thus $\mathcal{L}_{\mathcal{S}} = \rho^2\mu^2 - 2\rho\mu + (1 + \lambda\rho^2)$. Setting $\mathcal{L}_{\mathcal{S}} = 0$ gives a second order equation in ρ which does not have real-valued solutions for $\lambda > 0$. Thus in the presence of regularization, there exist no solutions that have the same margin for all points and reach zero empirical loss. However, solutions that have the same margin for all points and correspond to zero gradient w.r.t. ρ exist. To see this, assume $\sigma = 0$, setting the gradient of $\mathcal{L}_{\mathcal{S}}$ w.r.t. ρ equal to zero, yielding $\rho\mu^2 - \mu + \lambda\rho = 0$. This gives $\rho = \frac{\mu}{\mu^2 + \lambda}$. This solution yields $\rho\mu < 1$, which corresponds to non-interpolating solutions. ■

Figure 11 shows that the margins (which are never interpolating; interpolation is quivalent to $\rho y_n f(x_n) < 1, \quad \forall n$) tend to become equal to each other as predicted from the lemma during convergence.

Experiments We performed binary classification experiments using the standard CIFAR10 dataset (44). Image samples with class labels 1 and 2 were extracted for the binary classification task. The total number of training and test data points are 10000 and 2000, respectively. The model architecture in Fig. 1b contains four convolutional layers, two fully connected layers with hidden sizes 1024 and 2. The number of channels for the four convolutional layers are 32, 64, 128 and 128, the filter size is

[4]Notice that the term "margin" is usually defined as $\min_{n\in[N]} \bar{f}_n$. Instead, we use the term "margin for x_n" to distinguish our definition from the usual one.

3×3. The first fully connected layer has $3200 \times 1024 = 3,276,800$ weights and the very last layer has $1024 \times 2 = 2048$ weights. At the top layer of our model, there is a learnable parameter ρ (Fig. 1b). Instead of using Lagrange multipliers we could have used the equivalent (see (2)) Weight Normalization (WN), freezing the weights of the WN parameter "g" (45) in the Weight Normalization algorithm and normalizing the $\{V_k\}_{k=1}^{L-1}$ matrices at each layer w.r.t. their Frobenius norm, while the top layer's norm is denoted by ρ.

3.3 Landscape of the empirical risk

As a next step, we establish key properties of the loss landscape. The landscape of the empirical loss contains a set of degenerate zero-loss global minima (for $\lambda = 0$) that under certain overparametrization assumptions may be connected in a single zero-loss degenerate valley for $\rho \geq \rho_0$. Figure 2 shows a landscape which has a saddle for $\rho = 0$ and then goes to zero loss (zero crossing level, that is the coastline) for different values of ρ (look at the boundary of the mountain). As we will see in our analysis of the gradient flow, the descent from $\rho = 0$ can encounter local minima and saddles with non-zero loss. Furthermore, even though the valley of zero loss may be connected, the point of absolute minimum ρ may be unreachable by gradient flow from another point of zero loss even in the presence of $\lambda > 0$, because of the possible non-convex profile of the coastline (see inset of Figure 2).
If we assume overparametrized networks with $d \gg n$, where d is the number of parameters and N is the number of data points (10) proved that the global minima of the unregularized loss function $\mathcal{L}_S = \sum_{i=1}^{N}(f_W(x_i) - y_i)^2$ are highly degenerate[5] with dimension $d - N$.

Theorem 1 ((46), informal) *We assume an overparametrized neural network f_W with smooth ReLU activation functions and square loss. Then the minimizers W^* achieve zero loss and are highly degenerate with dimension $d - N$.*

Furthermore, for "large" overparametrization, all the global minima – associated to interpolating solutions – are connected within a unique, large valley. The argument is based on Theorem 5.1 of (47):

Theorem 2 ((47), informal) *If the first layer of the network has at least $2N$ neurons, where N is the number of training data and if the number of neurons in each subsequent layer decreases, then every sublevel set of the loss is connected.*

In particular, the theorem implies that *zero-square-loss minima with different values of ρ are connected*. A connected single valley of zero loss *does not* however guarantee that *SGD with WD will converge to the global minimum which is now > 0, independently of initial conditions*.
For large ρ we expect many solutions. The existence of several solutions for large ρ is based on the following intuition: the last linear layer is enough – if the layer before the linear classifier has more units than the number of training points – to provide solutions for a given set of random weights in the previous layers (for large ρ and small f_i). This also means that the intermediate layers do not need to change much under GD in the iterations immediately after initialization. The emerging picture is a landscape in which there are no zero-loss minima for ρ smaller than a certain minimum ρ, which is network and data-dependent. With increasing ρ from $\rho = 0$ there will be a continuous set of zero square-loss degenerate minima with the minimizer representing an interpolating (for $\lambda = 0$) or almost interpolating solution (for $\lambda > 0$). We expect that $\lambda > 0$ results in a "pull" towards the minimum ρ_0 within the local degenerate minimum of the loss.

Landscape for $\lambda > 0$ In the case of $\lambda\rho^2 > 0$ the landscape may become become a Morse-Bott or Morse function with shallow almost zero-loss minima. The question is open because the regularizer is not sum of squares (Yaim Cooper, personal communication).

[5]This result is also what one expects from Bezout theorem for a deep polynomial network. As mentioned in Terry Tao's blog "from the general "soft" theory of algebraic geometry, we know that the algebraic set V is a union of finitely many algebraic varieties, each of dimension at least $d - N$, with none of these components contained in any other. In particular, in the underdetermined case $N < d$, there are no zero-dimensional components of V , and thus V is either empty or infinite"(see references in (46)).

3.4 Gradient Dynamics

3.4.1 Gradient Flow Equations

The gradient flow equations are as follows (see also (8))

$$\begin{aligned}\dot{\rho} &= -\frac{\partial \mathcal{L}_{\mathcal{S}}(\rho, \{V_k\}_{k=1}^L)}{\partial \rho} = \frac{2}{N}\sum_n (1-\rho\bar{f}_n)\bar{f}_n - 2\lambda\rho \\ \dot{V}_k &= -\frac{\partial \mathcal{L}_{\mathcal{S}}(\rho, \{V_k\}_{k=1}^L)}{\partial V_k} = \frac{2}{N}\sum_n (1-\rho\bar{f}_n)\rho\frac{\partial \bar{f}_n}{\partial V_k} - 2\nu_k V_k.\end{aligned} \tag{2}$$

In the second equation we can use the unit norm constraint on the $\|V_k\|$ to determine the Lagrange multipliers ν_k, using the following structural property of the gradient:

Lemma 2 (Lemma 2.1 of (48)) *Let $f_W(x)$ be a ReLU neural network, $f_W(x) = W_L\sigma(W_{L-1}\dots\sigma(W_1 x)) : \mathbb{R}^d \to \mathbb{R}$. Then, we can write:*

$$\forall x \in \mathbb{R}^d : \ \sum_{i,j} \frac{\partial f_W(x)}{\partial W_k^{i,j}} W_k^{i,j} = \left\langle W_k, \frac{\partial f_W(x)}{\partial W_k} \right\rangle = f_W(x). \tag{3}$$

The constraint $\|V_k\|^2 = 1$ implies using the lemma above $\frac{\partial \|V_k\|^2}{\partial t} = V_k^T \dot{V}_k = 0$, which gives

$$\nu_k = \frac{1}{N}\sum_n (\rho\bar{f}_n - \rho^2 f_n^2) = \frac{1}{N}\sum_n \rho\bar{f}_n(1-\rho f_n). \tag{4}$$

Thus the gradient flow is the following dynamical system

$$\dot{\rho} = \frac{2}{N}\left[\sum_n \bar{f}_n - \sum_n \rho(\bar{f}_n)^2\right] - 2\lambda\rho \ \text{ and } \ \dot{V}_k = \frac{2}{N}\rho\sum_n \left[\left(1-\rho\bar{f}_n\right)\left(-V_k\bar{f}_n + \frac{\partial \bar{f}_n}{\partial V_k}\right)\right]. \tag{5}$$

In particular, we can also write

$$\dot{\rho} = 2(\mu - \rho(M+\lambda)), \tag{6}$$

hence, at critical points (when $\dot{\rho} = 0$ and $\dot{V}_k = 0$), we have

$$\rho = \rho_{\text{eq}} := \frac{\frac{1}{N}\sum_n \bar{f}_n}{\lambda + \frac{1}{N}\sum_n \bar{f}_n^2} = \frac{\sum_n \bar{f}_n}{\lambda + \sum_n \bar{f}_n^2} = \frac{\mu}{M+\lambda}. \tag{7}$$

Thus the gap to interpolation due to $\lambda > 0$ is $\epsilon = (\rho_{\lambda=0} - \rho_\lambda)\mu = 1 - \frac{\mu}{M+\lambda}\mu$ which gives

$$\epsilon = 1 - \frac{\mu^2}{\mu^2 + \sigma^2 + \lambda} = \frac{\sigma^2 + \lambda}{\mu^2 + \sigma^2 + \lambda}. \tag{8}$$

Notice that since the V_k are bounded functions they must take their maximum and minimum values on their compact domain – the sphere – because of the extremum value theorem. Also notice that for normalized V_k, $V_k^T\dot{V}_k = 0$ always, that is for normalized V_k the change in V_k is always orthogonal to V_k, that is V_k can only rotate. If $\dot{V}_k = 0$ then the weights V_k are given by[6]

$$V_k = \frac{\sum_n \ell_n \frac{\partial f_n}{\partial V_k}}{\sum \ell_n f_n}, \tag{9}$$

where $\ell_n = 1 - \rho\bar{f}_n$.

[6]This overdetermined system of equations – with as many equations as weights – can also be used to reconstruct the training set from the V_k, the y_n and the f_n.

Convergence. A favorable property of optimization of the square loss is the convergence of the relevant parameters. With gradient descent, the loss function cannot increase, while the trainable parameters may potentially diverge. A typical scenario of this kind happens with cross-entropy minimization, where the weights typically tend to infinity. In light of the theorems in Section 3.3, we could hypothetically think of training dynamics in which the loss function's value $\mathcal{L}(\rho, \{V_k\}_{k=1}^L)$ decreases while ρ oscillates periodically within some interval. As we show next, this is impossible when the loss function's value converges to zero.

Lemma 3 *Let $f_W(x) = \rho f_V(x)$ be a neural network and $\lambda = 0$. Assume that during training time, we have $\lim_{t\to\infty} \mathcal{L}(\rho, \{V_k\}_{k=1}^L) = 0$ and $\forall\ k \in [L] :\ \|V_k\| = 1$. Then, ρ and V_k converge (i.e., $\dot{\rho} \to 0$ and $\dot{V}_k \to 0$).*

Proof Note that if $\lim_{t\to\infty} \mathcal{L}(\rho, \{V_k\}_{k=1}^L) = 0$, then, for all $n \in [N]$, we have: $(\rho f_n - y_n)^2 \to 0$. In particular, $\rho f_n \to y_n$ and $\rho \bar{f}_n \to 1$. Hence, we conclude that $\mu\rho \to 1$. Therefore, by Lemma 4, $\rho\dot{\rho} \to 0$. We note that $\rho \to 0$ would imply $\rho f_n \to 0$ which contradicts $\mathcal{L}(\rho, \{V_k\}_{k=1}^L) \to 0$, since the labels y_n are non-zero. Therefore, we conclude that $\dot{\rho} \to 0$. To see why $\dot{V}_k \to 0$, we recall that

$$\dot{V}_k = \frac{2}{N}\rho \sum_n \left[\left(1 - \rho \bar{f}_n\right) \left(-V_k \bar{f}_n + \frac{\partial \bar{f}_n}{\partial V_k} \right) \right].$$

We note that $\|V_k\| = 1$, $|\bar{f}_n| = 1$ and $\frac{\partial \bar{f}_n}{\partial V_k}$ is bounded (assuming that $\forall\ n \in [N] :\ \|x_n\| \leq 1$ and $\forall\ k \in [L] :\ \|V_k\| = 1$). Hence, since ρ converges, $\rho \bar{f}_n \to 1$, implying (for $\lambda = 0$) $\dot{V}_k \to 0$. ■

So far, we have assumed convergence of GF, or GD or SGD to zero loss. Convergence does not seem too far fetched given overparametrization and the associated high degeneracy of the global minima (see 3.3 and theorems there). Proofs of convergence of descent methods have been however lacking until a recent paper (49) presented a new criterion for convergence of gradient descent and used to show that gradient descent with proper initialization converges to a global minimum. The result has technical limitations that are likely to be lifted in the future: it assumes that the activation function is smooth, that the input dimension is greater than or equal to the number of data points and that the descent method is GF or GD.

3.5 Qualitative Dynamics

We consider the dynamics of model b) in Figure 1. During training the norm of each layer weight matrix is kept constant by the Lagrange multiplier constraint which is applied to all layers but the last one, Thus leaving ρ at the top to change depending on the dynamics. Recall that $\forall\ n \in [N] :\ 0 \leq |\bar{f}_n| \leq 1$ because the assumption $\|x\| \leq 1$, yields $\|f(x)\| \leq 1$ by taking into account the definition of ReLUs and the fact that matrix norms are sub-multiplicative. Depending on the number of layers, the maximum margin that the network can achieve for a given dataset is usually much smaller than the upper bound 1, because the weight matrices have unit norm and the bound ≤ 1 is conservative. Thus, in order to guarantee interpolation, namely, $\rho f_n y_n = 1$, ρ must be significantly larger than 1. For instance, in the experiments plotted in this paper, the maximal $\bar{f}_n$ is ≈ 0.002 and thus the ρ needed for interpolation (for $\lambda = 0$) is in the order of 500. We assume then that for a given dataset there is a maximal value of $y_n f_n$ that allows interpolation. Correspondingly, there is a minimum value of ρ that we call, as mentioned earlier, ρ_0.
We now provide some intuition for the dynamics of the model. Notice that $\rho(t) = 0$ and $f_V(x) = 0$ (if all weights are zero) is a critical unstable point. A small perturbation will either result in $\dot{\rho} < 0$ with ρ going back to zero or in ρ growing if the average margin is just positive, that is $\mu > \lambda\rho > 0$.

Small ρ initialization. First, we consider the case where the neural network is initialized with a smallish ρ, that is $\rho < \rho_0$. Assume then that at some time t, $\mu > 0$, that is *average separability* holds. Notice that if the f_n were zero-mean, random variables, there would be a 50% chance for average separability to hold. Then Equation (5) shows that $\dot{\rho} > 0$. If full separability takes place, that is $\forall\ n :\ f_n > 0$, then $\dot{\rho}$ remains positive at least until $\rho = 1$. This is because Equation (5) implies that $\dot{\rho} \geq 2(\mu - \rho\mu)$ since $M \leq \mu$. In general, assuming eventual convergence, ρ may grow non-monotonically, that is there may oscillations in ρ for "short" intervals, until it converges to ρ_0.
To see this, consider the following lemma that gives a representation of the loss function in terms of ρ, $\dot{\rho}$ and μ.

Lemma 4 *Let $f_W(x) = \rho f_V(x)$ be a neural network, with $\forall\ k \in [L]:\ \|V_k\| = 1$. The square loss can be written as $\mathcal{L}_S(\rho, \{V_k\}_{k=1}^L) = 1 - \rho(\frac{1}{2}\dot{\rho} + \mu)$.*

Proof First, we consider that

$$\begin{aligned}\mathcal{L}_S(\rho, \{V_k\}_{k=1}^L) &= \frac{1}{N}\sum_n (\rho f_n - y_n)^2 + \sum_{k=1}^L \nu_k(\|V_k\|^2 - 1) + \lambda\rho^2 \\ &= \frac{1}{N}(\rho^2 f_n^2 - 2y_n\rho f_n + y_n^2) + \lambda\rho^2 \\ &= 1 - 2\rho\mu + \rho^2 M + \lambda\rho^2,\end{aligned} \tag{10}$$

where the second equations follows from $\forall\ k \in [L]:\ \|V_k\| = 1$ and the third from $y_n^2 = 1$, $\mu = \sum_n y_n f_n$ and $M = \sum_n f_n^2$. On the other hand, by Equation (6), $\dot{\rho} = 2\mu - 2\rho M - 2\lambda\rho$ which gives $2\rho M = 2\mu - 2\lambda\rho - \dot{\rho}$. Therefore, we conclude that $\mathcal{L}_S(\rho, \{V_k\}_{k=1}^L) = 1 - \frac{1}{2}\rho\dot{\rho} - \rho\mu = 1 - \rho(\frac{1}{2}\dot{\rho} + \mu)$ as desired. ■

Following this lemma, if $\dot{\rho}$ becomes negative during training, then, the average margin μ must increase since GD cannot increase but only decrease $\mathcal{L}$. In particular, this implies that $\dot{\rho}$ cannot be negative for long periods of time. Notice that short periods of decreasing ρ are "good" since they increase the average margin!

If $\dot{\rho}$ turns negative, it means that it has crossed $\dot{\rho} = 0$. This may be a critical point for the system if the values of V_k corresponding to $\dot{V}_k = 0$ are compatible (since the matrices $\{V_k\}_{k=1}^L$ determine the value of $\bar{f}_n$). We assume that this critical point – either a local minimum or a saddle – can be avoided by the randomness of SGD or by an algorithm that restarts optimization when a critical point is reached for which $\mathcal{L} > 0$.

Thus, ρ grows (non-monotonically) until it reaches an equilibrium value, close to ρ_0. Recall that for $\lambda = 0$ this corresponds to a degenerate global minimum $\mathcal{L} = 0$, usually resulting in a large attractive basin in the loss landscape. For $\lambda = 0$, a zero value of the loss ($\mathcal{L} = 0$) implies interpolation: thus all the f_n have the same value, that is all the margins are the same.

Large ρ initialization. If we initialize a network with large norm $\rho > \rho_0$, Equation (1) shows that $\dot{\rho} < 0$. This implies that the norm of the network will decreases until eventually an equilibrium is reached. In fact since $\rho \gg 1$ it is likely that there exists an interpolating (or near interpolating) solution with ρ that is very close to the initialization. In fact, for large ρ it is usually empirically possible to find a set of weights V_L such that $\rho\bar{f}_n \approx 1$. To understand why this may be true, recall that if there are at least N units in the top layer of the network (layer L) with given activities and $\rho \gg \rho_0$ there exist values of V_L that yield interpolation due to Theorem 2. In other words, it is easy for the network to interpolate with small values $\bar{f}_n$. These large ρ, small $\bar{f}_n$ solutions are reminiscent of the Neural Tangent Kernel (NTK) solutions (24), where the parameters do not move too far from their initialization. A formal version of the same argument is based on the following result.

We now assume that the network in the absence of weight decay has converged to an interpolating solution

Lemma 5 *Let f_V be a neural network with weights $\{V_k\}_{k=1}^L$, such that, $\forall\ n \in [N]:\ \rho\bar{f}_n = \rho\mu^* = 1$. Further assume that the classifier V_L and the last layer features h are aligned, ie, $y_n\langle V_L, h(x_n)\rangle = ||h(x_n)||_2$, where the vector h denotes the activities of the units in the last layer. Then, perturbing V_L into another unit-norm vector $V_L' \in \mathbb{R}^p$, such that, $V_L^T V_L' = \alpha \in (0, 1)$ yields a neural network $\hat{f}(x) = \langle V_L', h(x)\rangle$ with the property that $\frac{\rho}{\alpha}\hat{f}$ is an interpolating solution, corresponding to a critical point of the gradient but with a larger ρ.*

Proof Consider the margins of the network $\hat{f}(x) = \langle V_L', h(x)\rangle$, we have that $\bar{\hat{f}}_n = y_n\langle V_L', h(x_n)\rangle$. Since the classifier weights and the last layer features are aligned (as it may happen for $\lambda \to 0$), we have that $y_n h(x_n) = ||h(x_n)|| \times V_L$. This means $\bar{\hat{f}}_n = ||h(x_n)|| \times \langle V_L', V_L\rangle$. We also have from the interpolating condition that $\rho\bar{f}_n = \rho\mu^* = 1$, which means $||h(x_n)|| = \frac{1}{\rho}$. Putting all this together, we have $\frac{\rho}{\alpha}\bar{\hat{f}}_n = 1$, which concludes the proof. ■

Thus if a network exists providing an interpolating solution with a minimum ρ and $V_L \propto h$, there exist networks, that differ only in the last V_L layer, that are also interpolating but with larger ρ. As a consequence there is a continuum of solutions that differ only in the weights V_L of the last layer.

Of course there may be interpolating solutions corresponding to different sets of weights in layers below L, to which the above statement does not apply. These observations suggest that there is a valley of minimizers for increasing ρ, starting from a zero-loss minimizer which have the Neural Collapse property (see section 5).
In Figure 4 we show the dynamics of ρ alongside train loss and test error. We show results with and without Weight Decay in the top and bottom rows of Figure 4 respectively. $\mathcal{L}_S$ decreases with μ increasing and σ decreasing. The figures show that in our experiments the large margins of some of the data points decrease during GD, contributing to a decrease in σ. Furthermore Equation (10) suggests that for small ρ, the term dominating the decrease in $\mathcal{L}_S$ is $-2\rho\mu$. For larger ρ, the term $\rho^2 M = \rho^2(\sigma^2+\mu^2)$ becomes important: eventually $\mathcal{L}_S$ decreases because σ^2 decreases. The regularization term, for standard small values of λ, is relevant only in the final phase, when ρ is in the order of ρ_0. For $\lambda = 0$ the loss at the global equilibrium (which happens at $\rho = \rho_0$) is $\mathcal{L}_S = 0$ (since $\mu = \frac{1}{\rho_0}$, $M = \mu^2$, $\sigma^2 = 0$).
To sum up, starting from small initialization, gradient techniques will explore critical points with ρ growing from zero. Thus quasi-interpolating solutions with small ρ (corresponding to large margin solutions) may be found before the many large ρ quasi-interpolating solutions which have worse margins (see Figure 4, upper and lower row). This dynamics can take place *even in the absence of regularization*; however, $\lambda > 0$ makes the process more robust and bias it towards small ρ.

4 Generalization: Rademacher complexity of convolutional layers

4.1 Classical Rademacher bounds

In this section we analyze the test performance of the learned neural network. Following the standard learning setting, we assume that there is some underlying distribution P of labeled samples (x, y) and the training data $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^N$ consists of N i.i.d. samples from P. The model f_W is assumed to perfectly fit the training samples, i.e., $f_W(x_i) = y_i = \pm 1$.
We would like to upper bound the classification error $err(f_W) := \mathbb{E}_{(x,y)\sim P}[I[\text{sign}(f_W(x)) \neq y]]$ of the learned function f_W in terms of the number of samples N and the norm ρ of f_W.
This analysis is based on the following data-dependent measure of the complexity of a class of functions.

Definition 3 (Rademacher Complexity) *Let $\mathbb{H}$ be a set of real-valued functions $h : \mathcal{X} \to \mathbb{R}$ defined over a set $\mathcal{X}$. Given a fixed sample $S \in \mathcal{X}^m$, the empirical Rademacher complexity of $\mathbb{H}$ is defined as follows:*

$$\mathcal{R}_{\mathcal{S}}(\mathbb{H}) := \frac{2}{m}\mathbb{E}_\sigma \left[\sup_{h\in\mathbb{H}} \Big| \sum_{i=1}^m \sigma_i h(x_i) \Big| \right].$$

The expectation is taken over $\sigma = (\sigma_1, \ldots, \sigma_m)$, where, $\sigma_i \in \{\pm 1\}$ are i.i.d. and uniformly distributed samples.

The Rademacher complexity measures the ability of a class of functions to fit noise. The empirical Rademacher complexity has the added advantage that it is data-dependent and can be measured from finite samples.

Theorem 4 *Let P be a distribution over $\mathbb{R}^d \times \{\pm 1\}$. Let $\mathbb{F} = \{f_W \mid \prod_{i=1}^L \|W_i\| \leq 1\}$. Let $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^N$ be a dataset of i.i.d. samples selected from P. Then, with probability at least $1-\delta$ over the selection of $\mathcal{S}$, for any f_W that perfectly fits the data (i.e., $f_W(x_i) = y_i$), we have*

$$err_P(f_W) \leq 2(\rho+1) \cdot \mathcal{R}_{\mathcal{S}}(\mathbb{F}) + 3\sqrt{\frac{\log(2(\rho+1)^2/\delta)}{2N}} \tag{11}$$

Proof Let $t \in \mathbb{N} \cup \{0\}$ and $\mathbb{G}_t = \{f_W \mid \prod_{i=1}^L \|W_i\|_2 \in [t, t+1]\}$. We consider the ramp loss function

$$\ell_{ramp}(y, y') = \begin{cases} 1, & \text{if} \quad yy' \leq 0, \\ 1 - yy', & \text{if} \quad 0 \leq yy' \leq 1, \\ 0, & \text{if} \quad yy' \geq 1. \end{cases}$$

By (cf. (50), Theorem 3.3), for any $t \in \mathbb{N} \cup \{0\}$, with probability at least $1 - \frac{\delta}{t(t+1)}$, for any function $f_W \in \mathbb{G}_t$, we have

$$\mathbb{E}_{(x,y)}[\ell_{ramp}(f_W(x), y)] \leq \frac{1}{N}\sum_{i=1}^{N} \ell_{ramp}(f_W(x_i), y_i) + 2\mathcal{R}_{\mathcal{S}}(\mathbb{G}_t) + 3\sqrt{\frac{\log(2(t+1)^2/\delta)}{2N}}. \tag{12}$$

We note that for any function f_W for which $f_W(x_i) = y_i = \pm 1$, we have $\ell_{ramp}(f_W(x_i), y_i) = 0$. In addition, for any function f_W and pair (x, y), we have $\ell_{ramp}(f_W(x), y) \geq I[\text{sign}(f_W(x)) \neq y]$. Therefore, we conclude that with probability at least $1 - \frac{\delta}{t(t+1)}$, for any function $f_W \in \mathbb{G}_t$, we have

$$err_P(f_W) \leq 2\mathcal{R}_{\mathcal{S}}(\mathbb{G}_t) + 3\sqrt{\frac{\log(2(t+1)^2/\delta)}{2N}}. \tag{13}$$

We notice that by the homogeneity of ReLU neural networks, we have $\mathcal{R}_{\mathcal{S}}(\mathbb{G}_t) \leq (t+1) \cdot \mathcal{R}_{\mathcal{S}}(\mathbb{F})$. By union bound over all $t \in \mathbb{N} \cup \{0\}$, (13) holds uniformly for all $t \in \mathbb{N} \cup \{0\}$ and $f_W \in \mathbb{G}_t$ with probability at least $1 - \delta$. For each f_W with $\prod_{i=1}^{L} \|W_i\|_2 = \rho$ we can apply the bound with $t = \lfloor \rho \rfloor$ since $f_W \in \mathbb{G}_t$, and obtain the desired bound,

$$\begin{aligned} err_P(f_W) &\leq 2(t+1) \cdot \mathcal{R}_{\mathcal{S}}(\mathbb{G}_t) + 3\sqrt{\frac{\log(2(t+1)^2/\delta)}{2N}} \\ &\leq 2(\rho+1) \cdot \mathcal{R}_{\mathcal{S}}(\mathbb{F}) + 3\sqrt{\frac{\log(2(\rho+1)^2/\delta)}{2N}} \end{aligned} \tag{14}$$

■

The above theorem provides an upper bound on the classification error of the trained network f_W that perfectly fits the training samples. The upper bound is decomposed into two main terms. The first term is proportional to the norm of the trained model ρ and the Rademacher complexity of $\mathbb{F}$ which is the set of the normalized neural networks and the second term scales as $\sqrt{\log(\rho/\delta)/N}$. As shown in Theorem 1 in (51), this term is upper bounded by $\mathcal{R}_{\mathcal{S}}(\mathbb{F}) \leq (\sqrt{2\log(2)L} + 1)/\sqrt{(N)}$, assuming that the samples are taken from the d-dimensional ball $\mathbb{B}_d$ of radius 1. The overall bound is then (assuming zero training error)

$$err_P(f_W) \leq \frac{2(\rho+1)(\sqrt{2\log(2)L}+1)}{\sqrt{N}} + 3\sqrt{\frac{\log(2(\log(\rho)+1)^2/\delta)}{2N}}. \tag{15}$$

We note that while the mentioned bound on $\mathbb{R}_N(\mathbb{F})$ depends on the architecture of the network it does not depend in an explicit way on the training set. However, as shown in Equation 6 in (51), the bound may be improved further if the matrices' stable rank is low, which happens with small rank of the weight matrices. In practice, the value of $\mathbb{R}_N(\mathbb{F})$ depends on the network architecture (e.g. convolutional) but also on the underlying optimization (e.g. L_2 vs L_1) and on the data (e.g. rank).

4.2 Relative Generalization

We now consider two solutions with zero empirical loss of the square loss regression problem obtained with the *same* ReLU deep network and corresponding to two different minima with two different ρ. Let us call them $g^a(x) = \rho_a f^a(x)$ and $g^b(x) = \rho_b f^b(x)$. Using the notation of this paper, the functions f_a and f_b correspond to networks with normalized weight matrices at each layer.
Let us assume that $\rho_a < \rho_b$.
We now use Equation 15 and the fact that the empirical $\hat{L_\gamma}$ *for both functions is the same* to write $L_0(f^a) = L_0(F^a) \leq c_1\rho_a\mathbb{R}_N(\tilde{\mathbb{F}}) + c_2\sqrt{\frac{\ln(\frac{1}{\delta})}{2N}}$ and $L_0(f^b) = L_0(F^b) \leq c_1\rho_b\mathbb{R}_N(\tilde{\mathbb{F}}) + c_2\sqrt{\frac{\ln(\frac{1}{\delta})}{2N}}$. The bounds have the form

$$L_0(f^a) \leq A\rho_a + \epsilon \tag{16}$$

and

$$L_0(f^b) \leq A\rho_b + \epsilon \tag{17}$$

Thus the upper bound for the expected error $L_0(f^a)$ is better than the bound for $L_0(f^b)$. Of course this is just an upper bound. As a consequence this result does not guarantee that a solution with smaller ρ will always have a smaller expected error than a solution with larger ρ.
Notice that the this generalization claim is just a *relative* claim about different solutions obtained with the same network trained on the same training set.
Figure 5 shows clearly that increasing the percentage of random labels increases the ρ that is needed to maintain interpolation – thus decreasing the margin – and that at the same time the test error increases, as expected. This monotonic relation between margin and accuracy at test seems to break down for small differences in margin as shown in Figure 6, though the significance of the effect is unclear. Of course this kind of behavior is not inconsistent with an upper bound.

4.3 Novel bounds for Sparse Networks

old versionThe bounds in section 4.1 are for generic deep networks. In such a general case, ρ in those bounds is the product of the Frobenius norms of all the weight matrices. For convolutional networks the weight matrices are Toeplitz matrices. Thus a direct application of Equation 15 gives large bounds – though not as large as in the case of dense networks (because of the sparsity of the Toeplitz matrices). Here we show that the bound on the Rademacher complexity can be reduced by exploiting two typical properties of CNNs: a) the locality of the convolutional kernels and b) shared weights. They allow us to use only the norm of the kernels in the calculation of ρ_k instead of the norm of the corresponding Toeplitz matrix. In this section we give an outline of the results with more precise statements and proofs to be published later.
We start by considering the simple situation of non-overlapping convolutional patches. In other words, the stride of the convolution is equal to the size of the kernel in each layer. This means that in the associated Toeplitz matrix the non-zero components in each row do not overlap with the non-zero components of the row above or the one below. In other words, if K is the number of patches, ℓ is the size of each patch and $x \in \mathbb{R}^d$, then $d = K\ell$. Notice that the standard bounds give a Rademacher complexity proportional to the product of the Frobenius norms of each weight matrix $\|W\|$ time the norm of $\|x\|$, where $\|W\| \propto \sqrt{k}M$, where M is the norm of the kernel. In section 4.1 we describe generic bounds on the Rademacher complexity of deep neural networks. In these cases, ρ measures the product of the Frobenius norms of the network's weight matrices in each layer. For convolutional networks, however, the operation in each layer is computed with a kernel, described by the vector w, that acts on each patch of the input separately. Therefore, a convolutional layer is represented by a Toeplitz matrix W, whose blocks are each given by w. A naive application of (15) to convolutional networks give a large bound, where the Frobenius norm of the Toeplitz matrix is equivalent to norm of the kernel multiplied by the number of patches.
In this section we provide an informal analysis of the Rademacher complexity, showing that it can be reduced by exploiting maonly the first one of the two properties of convolutional layers: (a) the locality of the convolutional kernels and (b) weight sharing. These properties allow us to bound the Rademacher complexity by taking the products of the norms of the kernel w instead of the norm of the associated Toeplitz matrix W. Here we outline the results with more precise statements and proofs to be published separately.
We consider the case of 1-dimensional convolutional networks with non-overlapping patches and one channel per layer. For simplicity, we assume that the input of the network lies in $\mathbb{R}^d$, with $d = 2^L$ and the stride and the kernel of each layer are 2. The analysis can be easily extended to kernels of different sizes. This means that the network $h(x)$ can be represented as a binary tree, where the output neuron is computed as $W^L \cdot \sigma(v_1^L(x), v_2^L(x))$, $v_1^L(x) = W^{L-1} \cdot \sigma(v_1^{L-1}(x), v_2^{L-1}(x))$ and $v_2^L(x) = W^{L-1} \cdot \sigma(v_3^{L-1}(x), v_4^{L-1}(x))$ and so on. This means that we can write the i'th row of the Toeplitz matrix of the l'th layer $(0, \ldots, 0, -W^l-, 0 \ldots, 0)$, where W^l appears on the $2^i - 1$ and 2^i coordinates. We define a set $\mathcal{H}$ of neural networks of this form, where each layer is followed by a ReLU activation function and $\prod_{l=1}^{L} W^l \leq \rho$.

Theorem 5 *Let $\mathcal{H}$ be the set of binary-tree structured neural networks over $\mathbb{R}^d$, with $d = 2^L$ for some natural number L. Let $X = \{x_1, \ldots, x_m\} \subset \mathbb{R}^d$ be a set of samples. Then,*

$$\mathcal{R}_X(\mathcal{H}) \leq \frac{2^L \rho \sqrt{\sum_{i=1}^{m} \|x_i\|^2}}{m} \tag{18}$$

Proof [Proof sketch] First we rewrite the Rademacher complexity in the following manner:

$$\begin{aligned}\mathcal{R}_X(\mathcal{H}) &= \mathbb{E}_\epsilon \sup_{h\in\mathcal{H}} |\frac{1}{m}\sum_{i=1}^{m}\epsilon_i \cdot h(x_i)| \\ &= \mathbb{E}_\epsilon \sup_{h\in\mathcal{H}} \frac{1}{m}|\sum_{i=1}^{m}\epsilon_i \cdot W^L \cdot \sigma(v_1(x), v_2(x))| \\ &= \mathbb{E}_\epsilon \sup_{h\in\mathcal{H}} \frac{1}{m}\sqrt{|\sum_{i=1}^{m}\epsilon_i \cdot W^L \cdot \sigma(v_1(x), v_2(x))|^2}\end{aligned} \tag{19}$$

Next, by the proof of Lem. 1 in (51), we obtain that

$$\begin{aligned}\mathcal{R}_X(\mathcal{H}) &\leq 2\mathbb{E}_\epsilon \sup_{h\in\mathcal{H}} \frac{1}{m}\sqrt{\|W^L\|^2 \cdot \|\sum_{i=1}^{m}\epsilon_i(v_1(x), v_2(x))\|^2} \\ &= \mathbb{E}_\epsilon \sup_{h\in\mathcal{H}} \frac{1}{m}\sqrt{\|W^L\|^2 \cdot \sum_{j=1}^{2}\|\sum_{i=1}^{m}\epsilon_i v_j(x_i)\|^2}\end{aligned} \tag{20}$$

By applying this peeling process L times, we obtain the following inequality:

$$\begin{aligned}\mathcal{R}_X(\mathcal{H}) &\leq 2^{L-1}\mathbb{E}_\epsilon \sup_{h\in\mathcal{H}} \frac{1}{m}\sqrt{\prod_{l=1}^{L}\|W^l\|^2 \cdot \sum_{j=1}^{d}\|\sum_{i=1}^{m}\epsilon_i x_{ij}\|^2} \\ &= 2^{L-1}\mathbb{E}_\epsilon \sup_{h\in\mathcal{H}} \frac{1}{m}\sqrt{\prod_{l=1}^{L}\|W^l\|^2 \cdot \|\sum_{i=1}^{m}\epsilon_i x_i\|^2} \\ &\leq \frac{2^{L-1}\rho\mathbb{E}_\epsilon\|\sum_{i=1}^{m}\epsilon_i x_i\|}{m} \\ &\leq \frac{2^{L-1}\rho\sqrt{\sum_{i=1}^{m}\|x_i\|^2}}{m}\end{aligned} \tag{21}$$

where the factor 2^{L-1} is obtained because the last layer is linear (see (52)). We note that a better bound can achieved when using the reduction introduced in (51) which would give a factor of $\sqrt{2\log(2)L}+1$ instead of 2^{L-1}. ■

One-layer convolutional classifier Consider a ReLU convolutional classifier with k patches. $\hat{\mathcal{R}}_m$, in the standard bounds would be

$$\hat{\mathcal{R}}_m \leq BX$$

where B is the Frobenius norm of the Toeplitz matrix with k rows, each row consisting of the kernel w. Thus $B = \sqrt{K}\|w\|$ and $X = \|x\|$.
Our calculation gives with x^1 representing the first patch of x and x^K the last one:

$$\hat{\mathcal{R}}_m \leq \sqrt{\|w\|^2\|x^1 + \cdots + x^K\|^2} = \sqrt{\|w\|^2\|x\|^2} = \|w\|\|x\|.$$

instead of the general bound usually referred which is

$$\hat{\mathcal{R}}_m \leq \|W\|\|x\| = \sqrt{k}\|w\|\|x\|$$

Multi-layer convolutional classifier The Rademacher complexity of a feed-forward neural network can be bounded recursively by considering each layer at a time. A bound that can be used for the recursion is given by the following proposition (see (52; 51)), that expresses the Rademacher complexities at the outputs of one layer in terms of the outputs at the previous layers.

Lemma 6 *Let $\mathcal{H}$ be a class of functions from $\mathbb{R}^d$ to $\mathbb{R}$. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be the ReLU function which is 1-Lipschitz and define $\mathcal{H}' := \left\{x \in \mathbb{R}^d \to \sigma\left(\sum_{j=1}^k w_j h_j(x)\right) \in \mathbb{R} : \|w\|_2 \leq M, h_1, \ldots, h_k \in \mathcal{H}\right\}$. Then, for any $x_1, \ldots, x_m \in \mathbb{R}^d$*

$$\mathcal{R}\left(\mathcal{H}' \circ \{x_1, \ldots, x_m\}\right) \leq 2M\mathcal{R}\left(\mathcal{H} \circ \{x_1, \ldots, x_m\}\right).$$

We apply now the Lemma to the class of L-depth ReLU real-valued CNN, with each layer's kernel w_d with norm at most M_d.

Theorem 6 (*informal*) *The Rademacher complexity of a convolutional deep net with ReLUs in all d layers but the last linear one and with non-overlapping convolutional patches can be bounded as*

$$\hat{\mathcal{R}_m(\mathcal{H}_d)} \leq (\sqrt{2\log(2)L} + 1) \prod_{j=1}^{L} M_j \|x\| \tag{22}$$

Proof [Proof sketch] Each $h_k^d \in \mathcal{H}_\lceil$ $(k = 1, \cdots, Q$ is a ReLU classifier inputs from patch j of the layer below. Patch k in layer $d-1$ can be written as a vector v_k consisting of ℓ classifiers $v_k = h_{k\cdot 1}^{d-1}, h_{k\cdot 2}^{d-1}, \cdots, h_{k\cdot \ell}^{d-1}$. Then $h_k^d = \sigma(w \cdot v_k$. Notice that because of our assumption of non-overlapping patches there number of units in layer $d-1$ is ℓ times the number of units in layer d. Then

$$\hat{\mathcal{R}}_m\left(\mathcal{H}_d\right) = \mathbb{E}_\epsilon \sup_{h_i \in \mathcal{H}_d} \frac{1}{m} \sum_{i=1}^{m} \epsilon_i h_i = \mathbb{E}_\epsilon \sup_{h_i \in \mathcal{H}_{d-1} w: \|w\| \leq M} \frac{1}{m} \sum_{i=1}^{m} \epsilon_i w \cdot (\sum_k v_k), \tag{23}$$

can be upper bounded as follows

$$\hat{\mathcal{R}}_m\left(\mathcal{H}_d\right) \leq 2M_d \mathbb{E}_\epsilon \sup_{h \in \mathcal{H}_d} \|\frac{1}{m} \sum_{i=1}^{m} \epsilon_i (\sum_k (v_k)_i)\| \leq \frac{1}{m} \sqrt{(w \cdot (\sum_k v_k)^2} = \frac{1}{m} \sqrt{(w \cdot (\sum_k v_k)^2} = 2M_d \hat{\mathcal{R}}_m\left(\mathcal{H}_{d-1}\right), \tag{24}$$

because $(\sum_k v_k)^2 = \sum_k v_k^2$ since the various patches are zero-mean and uncorrelated.
Continuing the peeling we obtain

$$\mathcal{R}_m(\mathcal{H}_L) \leq 2^{L-1} \hat{M_L} \cdot M_{L-1} \cdots M_1 \|x\|, \tag{25}$$

where the factor 2^{L-1} is obtained because the last layer is linear (see (52)). To this result we can further apply the reduction used by (51) to finally obtain the result. ■

One ends up with a bound scaling as the product of the norms of the kernel at each layer. The constants may change depending on the architecture, the number of patches, the size of the patches and their overlap.
Thus one ends up with a bound scaling as the product of the norms of the kernel at each layer. The constants may change depending on the architecture, the number of patches, the size of the patches and their overlap.
This special non-overlapping case can be extended to the general convolutional case. In fact a proof of the following conjecture will be provided in (53)

Conjecture 1 *If a convolutional layer has overlap among its patches then the non-overlap bound*

$$\mathcal{R}_m(\mathcal{H}_L) \hat{\leq} 2^{L-1} \rho \|x\| \tag{26}$$

where ρ is the product of the norms of the kernels at each layer becomes

$$\hat{\mathcal{R}_m(\mathcal{H}_L)} \leq 2^{L-1} \rho \sqrt{\frac{K}{K-O}} \|x\| \tag{27}$$

where K is the size of the kernel (number of components) and O is the size of the overlap.

Sketch proof Call P the number of patches and O the overlap. With no overlap then $PK = D$ where D is the dimensionality of the input to the layer. In general $P = \frac{D-O}{K-O}$. It follows that a layer with the most overlap can add at most $< \|x\|\sqrt{K}$ to the bound. Notice that we assume that each component of x_i averaged across i will have norm $\sqrt{\frac{1}{d}}$.

The bound is surprisingly small In this section we have derived bounds for convolutional networks that may potentially be orders of magnitude smaller than equivalent similar bounds for dense networks. We note that a naive application of Corollary 2 in (29) for the network we used in Theorem 5 would require treating the network as if it were a dense network. In this case the bound would be proportional to the product of the norms of each of the Toeplitz matrices in the network individually. In this case, the total bound becomes

$$\frac{2^L\sqrt{\prod_{l=1}^{L}(2^l)}\rho\sqrt{\sum_{i=1}^{m}\|x_i\|^2}}{m} = \frac{2^{0.25L^2+1.25L}\rho\sqrt{\sum_{i=1}^{m}\|x_i\|^2}}{m} \tag{28}$$

which is much larger than the bound we obtained earlier. The key point is that *the Rademacher bounds achievable for sparse networks are much smaller than for dense networks*. This suggests that convolutional network with local kernels may generalize much better than dense network, which is consistent *in spirit* with approximation theory results (compositionally sparse target functions can be approximated by sparse networks without incurring in the curse of dimensionality, whereas generic functions cannot be approximated by dense networks without the curse). They also confirm the empirical success of convolutional networks compared to densely connected networks.
It is also important to observe that the bounds we obtained may be non-vacuous in the overparametrized case, unlike VC bounds which depend on the number of weights and are therefore always vacuous in overparametrized situations. With our norm-based bounds it is in principle possible to have *over-parametrization and interpolation simultaneously with non-vacuous generalization bounds*: this is suggested by Figure 8. Figure 9 shows the case of a 3-layer convolutional network with a total number of parameters of $\approx 20K$.

5 Neural Collapse

A recent paper (12) described four empirical properties of the terminal phase of training (TPT) deep networks, using the cross-entropy loss function. TPT begins at the epoch where training error first vanishes. During TPT, the training error stays effectively zero, while training loss progressively decreases. Direct empirical measurements expose an inductive bias they call Neural Collapse (NC), involving four interconnected phenomena. Informally, (NC1) Cross-example within-class variability of last-layer training activations collapses to zero, as the individual activations themselves collapse to their class means. (NC2) The class means collapse to the vertices of a simplex equiangular tight frame (ETF). (NC3) Up to rescaling, the last-layer classifiers collapse to the class means or in other words, to the simplex ETF (i.e., to a self-dual configuration). (NC4) For a given activation, the classifier's decision collapses to simply choosing whichever class has the closest train class mean (i.e., the nearest class center decision rule).
We now formally define the four Neural Collapse conditions. We consider a network $f_W(x) = W_L h(x)$, where $h(x) \in \mathbb{R}^p$ denotes the last layer feature embedding of the network, and $W_L \in \mathbb{R}^{C\times p}$ contains the parameters of the classifier. The network is trained on a C-class classification problem on a balanced dataset $\mathcal{S} = \{(x_{cn}, y_{cn})\}_{n=1,c=1}^{N,C}$ with N samples per class. We can compute the per-class mean of the last layer features as follows

$$\mu_c = \frac{1}{N}\sum_{n=1}^{N} h(x_{cn}), \tag{29}$$

The global mean of all features as follows

$$\mu_G = \frac{1}{C}\sum_{c}\mu_c = \frac{1}{NC}\sum_{c=1,n=1}^{C,N} h(x_{cn}).$$

Furthermore, the second order statistics of the last layer features are computed as:

$$\Sigma_W = \frac{1}{C}\sum_{c=1}^{C}\frac{1}{N}\sum_{n=1}(h(x_{cn})-\mu_c)(h(x_{cn})-\mu_c)^\top$$
$$\Sigma_B = \frac{1}{C}\sum_{c=1}^{C}(\mu_c-\mu_G)(\mu_c-\mu_G)^\top \tag{30}$$
$$\Sigma_T = \frac{1}{NC}\sum_{c=1,n=1}^{C,N}(h(x_{cn})-\mu_G)(h(x_{cn})-\mu_G)^\top.$$

Here, Σ_W measures the within-class-covariance of the features, Σ_B is the between-class-covariance, and Σ_T is the total covariance of the features ($\Sigma_T = \Sigma_W + \Sigma_B$).
We can now list the formal conditions for Neural Collapse:

NC1 (Variability collapse) Variability collapse states that the variance of the feature embeddings of samples from the same class tends to zero, or formally, $\text{Tr}(\Sigma_W) \to 0$.

NC2 (Convergence to Simplex ETF) $|\|\mu_c-\mu_G\|_2 - \|\mu_{c'}-\mu_G\|_2| \to 0$, or the centered class means of the last layer features become equinorm. Moreover, if we define $\tilde{\mu}_c = \frac{\mu_c-\mu_G}{\|\mu_c-\mu_G\|_2}$, then we have $\langle\tilde{\mu}_c, \tilde{\mu}_{c'}\rangle = -\frac{1}{C-1}$ for $c \neq c'$, or the centered class means are also equiangular. The equinorm condition also implies that $\sum_c \tilde{\mu}_c = 0$, i.e., the centered features lie on a simplex.

NC3 (Self-Duality) If we collect the centered class means into a matrix $M = [\mu_c - \mu_G]$, we have $\left|\left|\frac{W^\top}{\|W\|_F} - \frac{M}{\|M\|_F}\right|\right| \to 0$, or that the classifier W and the last layer feature means M become duals of each other.

NC4 (Nearest Center Classification) The classifier implemented by the deep network eventually boils down to choosing the closest mean last layer feature $\text{argmax}_c\langle W_L^c, h(x)\rangle \to \text{argmin}_c\|h(x)-\mu_c\|_2$.

Related Work on Neural Collapse: Since the empirical observation of Neural Collapse was made in (12), a number of papers have studied the phenomenon in the so-called *Unconstrained Features* regime (32; 34; 40; 33; 39). The basic assumption underlying these proofs is that the features of a deep network at the last layer can essentially be treated as free optimization variables, which converts the problem of finding the parameters of a deep network that minimize the training loss, into a matrix factorization problem of factoring one-hot class labels $Y \in \mathbb{R}^{C\times CN}$ into the last layer weights $W \in \mathbb{R}^{C\times p}$ and the last layer features $H \in \mathbb{R}^{p\times CN}$. In the case of the squared loss, the problem that they study is $\min_{W,H}\|WH-Y\|^2 + \lambda_W\|W\|^2 + \lambda_H\|H\|^2$.
In this section, we show instead that we can predict the existence of Neural Collapse and its properties as a consequence of our analysis of the dynamics of SGD on deep binary classifiers trained on the square loss function with Lagrange Normalization and Weight Decay *without any additional assumption*. We first consider the case of binary classification and show that NC follows from the analysis of the dynamics of the square loss in the previous sections. The loss function is the same one defined in Equation (1), and we consider minimization using SGD with a batch size of 1. After establishing Neural Collapse in this familiar setting, we consider the multiclass setting where we derive the conditions of Neural Collapse from an analysis of the squared loss function with weight decay and weight normalization.

5.1 Binary Classification

We prove in this section that Neural Collapse follows from the following property of the landscape of the squared loss that we analyzed in the previous section:
[Symmetric Quasi-interpolation (Binary Classification)] Consider a binary classification problem with inputs in a feature space $\mathcal{X}$ and labels space $\{+1,-1\}$. A classifier $f_W : \mathcal{X} \to \mathbb{R}$ symmetrically quasi-interpolates a training dataset $\mathcal{S} = \{(x_n, y_n)\}_{n=1}^N$ if for all training examples $f_{W\,n} = y_n f_W(x_n) = 1-\epsilon$, where ϵ is the interpolation gap.

We prove first that the property *follows without any assumption* at convergence from our previous analysis of the landscape of the squared loss for binary classification.

Lemma 7 *An overparameterized deep ReLU network for binary classification trained to convergence under the squared loss in the presence of weight decay and weight normalization (WN) satisfies the symmetric quasi-interpolation property. Furthermore, the gap to interpolation of the regularized network is* $\epsilon = \frac{\lambda}{\mu^2+\lambda}$ *where* $\mu = \frac{1}{N}\sum_i \bar{f}_i$.

Proof
Consider the regularized square loss $\mathcal{L}_S = \frac{1}{N}\sum_{i=1}^{N}(\rho\bar{f}_i - 1)^2 + \lambda\rho^2$. We recall the definitions made earlier in section 3.2 of the margin $\bar{f}_i = y_i f_i$, and the first and second order sample statistics of the margin $\mu = \frac{1}{N}\sum_{i=1}^{N}\bar{f}_i, M = \frac{1}{N}\sum_{i=1}^{N}\bar{f}_i^2, \sigma^2 = M - \mu^2$. We consider deep networks that are sufficiently overparameterized to attain 100% accuracy on the training dataset, which means $\bar{f}_i > 0$. Since the weights of the deep network $\{V_k\}_{k=1}^{L}$ are normalized and the data x_i lie within the unit norm ball, we have that $|\bar{f}_i| \leq 1$. Even though $\bar{f}_i$ could take values close to 1, the typically observed values of $\bar{f}_i$ in our experiments are approximately 5×10^{-3}. For our purposes it suffices to note that there exists a maximum possible margin such that $0 < \bar{f}_i \leq \bar{\mu}$ for all training examples for a given data set and network architecture.
Using these definitions, we can rewrite the deep network training problem as:

$$\min_{\rho,\{V_k\}_{k=1}^{L}} \mathcal{L}_S = (M+\lambda)\rho^2 - 2\rho\mu + 1 \tag{31}$$

All critical points (including minima) need to satisfy $\frac{\partial \mathcal{L}_S}{\partial \rho} = 0$, from which we get $\rho = \frac{\mu}{M+\lambda}$. If we plug this back into the loss, our minimization problem becomes:

$$\begin{aligned}
&\min_{\{V_k\}_{k=1}^{L}} (M+\lambda)\left(\frac{\mu}{M+\lambda}\right)^2 - 2\frac{\mu^2}{M+\lambda} + 1 \\
&= \min_{\{V_k\}_{k=1}^{L}} 1 - \frac{\mu^2}{M+\lambda} \\
&= \min_{\{V_k\}_{k=1}^{L}} \frac{\sigma^2+\lambda}{\mu^2+\sigma^2+\lambda} \\
&= \min_{\{V_k\}_{k=1}^{L}} \frac{1}{1+\frac{\mu^2}{\sigma^2+\lambda}}
\end{aligned} \tag{32}$$

Hence in order to minimize the loss we have to find $\{V_k\}_{k=1}^{L}$ that maximize μ^2 and minimize σ^2. Since we assumed that the network is expressive enough to attain any value, the loss is minimized when $\sigma^2 = 0$ and $\mu = \bar{\mu}$. Thus all training examples have the same margin.
If $\sigma^2 \to 0$, then all margins tend to the same value, $\bar{f}_i \to \bar{\mu}$, and the optimum value of ρ is $\rho = \frac{\bar{\mu}}{\bar{\mu}^2+\lambda}$. This means that the gap to interpolation is $\epsilon = 1 - \rho\bar{\mu} = \frac{\lambda}{\lambda+\bar{\mu}^2}$.

■

The prediction $\sigma \to 0$ has empirical support: we show in Figure 11 that all the margins converge to be roughly equal. Once within class variability disappears, and for all training samples, the last layer features collapse to their mean. The outputs and margins then also collapse to the same value. We can see this in the left plot of Figure 13 where all of the margin histograms are concentrated around a single value. We visualize the evolution of the training margins over the training epochs in Figure 11 which shows that the margin distribution concentrates over time. At the final epoch the margin distribution (colored in yellow) is much narrower than at any intermediate epochs. Notice that our analysis of the origin of the SGD noise shows that "strict" NC1 never happens with SGD, in the sense that the margins are never, not even asymptotically, exactly equal to each other, but just very close!
We now prove that Neural Collapse follows from property 5.1.

Theorem 7 *Let* $\mathcal{S} = \{(x_n, y_n)\}_{n=1}^{N}$ *be a dataset. Let* (ρ, V) *be the parameters of a ReLU network* f *such that* V_L *has converged when training using SGD with batches of size* 1 *on the square loss with LN+WD. Let* $\mu_+ = \frac{1}{N}\sum_{n=1,y_n=1}^{N} h(x_n)$, $\mu_- = \frac{1}{N}\sum_{n=1,y_n=-1}^{N} h(x_n)$. *Consider points of convergence of SGD that satisfy Property 5.1. Those points also satisfy the conditions of Neural Collapse as described below.*

- *NC1:* $\mu_+ = h(x_n)$ *for all* $n \in [N], y_n = 1$, $\mu_- = h(x_n)$ *for all* $n \in [N], y_n = -1$;
- *NC2:* $\mu_+ = -\mu_-$, *which is the structure of an ETF with two vectors;*
- *NC3:* $V_L \propto \mu_+, \mu_-$;
- *NC4:* $sign(\rho f_V(x)) = \arg\min_{c \in \{+1,-1\}} \|\mu_c - h(x)\|$.

Proof The update equations for SGD on the square loss function with LN+WD are given by:

$$\begin{aligned} V_L^{(t+1)} &= V_L^{(t)} - \eta \frac{\partial \mathcal{L}}{\partial V_L^{(t)}} \\ \implies V_L^{(t+1)} &= V_L^{(t)} - \eta \times \left(2\rho(\rho \bar{f}_n - 1) y_n h(x_n) + 2\nu_L^{(t)} V_L^{(t)} \right) \end{aligned} \tag{33}$$

We can apply the unit norm constraints $||V_L^{(t+1)}||^2 = 1$ and $||V_L^{(t)}||^2 = 1$ and ignore all terms that are $O\left(\eta^2\right)$ to compute $\nu_L^{(t)}$ as:

$$\begin{aligned} 2\nu_L^{(t)} &= 2\rho y_n V_L^{(t)\top} h(x_n)(1 - \rho \bar{f}_n) \\ \implies \nu_L^{(t)} &= \rho \bar{f}_n (1 - \rho \bar{f}_n) \end{aligned} \tag{34}$$

This gives us the following SGD update:

$$V_L^{(t+1)} = V_L^{(t)} - \eta \times 2\rho y_n (\rho \bar{f}_n - 1) \left(h(x_n) - f_n V_L^{(t)} \right) \tag{35}$$

Using property 5.1, we can see that for every training sample in class $y_n = 1$, $h(x_n) = \frac{(1-\epsilon)}{\rho} V_L$, and for every training sample in class $y_n = -1$, $h(x_n) = \frac{(-1+\epsilon)}{\rho} V_L$. This shows that within class variability has collapsed and that all last layer features collapse to their mean, which is the condition for NC1. We can also see that $\mu_+ = -\mu_-$, which is the condition for NC2 when there are 2 vectors in the Simplex ETF. The SGD convergence condition also tells us that $V_L \propto \mu_+$ and $V_L \propto \mu_-$, which gives us the NC3 condition. NC4 follows then from NC1-NC2, as shown by theorems in (12) ■

5.2 Multiclass Classification

In the previous section we proved the emergence of Neural Collapse in the case of a binary classifier with scalar outputs, in order to be consistent with our framework in section 3. The phenomenon of Neural Collapse was however defined in (12) for the case of multiclass classification with deep networks. In this section we describe how NC emerges in this setting from the minimization of the squared loss with Weight Normalization and Weight Decay regularization.

We consider a classification problem with C classes with a balanced training dataset $\mathcal{S} = \cup_{c=1}^{C} \mathcal{S}_c = \cup_{c=1}^{C} \{(x_{cn}, c)\}_{n=1}^{N} = \{(x_n, y_n)\}$ that has N training examples $\mathcal{S}_c = \{(x_{cn}, c)\}_{n=1}^{N}$ per-class $c \in [C]$. The labels are represented by the unit vectors $\{e_c\}_{c=1}^{C}$ in $\mathbb{R}^C$. Since we consider deep homogeneous networks that do not have bias vectors, we center the one-hot labels, and scale them so that they have maximum output 1. We denote the resulting labels (for a class-balanced dataset) as $\tilde{e}_c = \left[\frac{-1}{C-1}, \ldots, \frac{-1}{C-1}, 1, \frac{-1}{C-1}, \ldots, \frac{-1}{C-1} \right]$, where the c^{th} coordinate is 1. We consider a deep ReLU network $f_W : \mathbb{R}^d \to \mathbb{R}^C$, which takes the form $f_W(x) = W_L \sigma(W_{L-1} \ldots W_2 \sigma(W_1 x) \ldots)$. However, we stick to the normalized reparameterization of the deep ReLU network as $f(x) = \rho V_L \sigma(V_{L-1} \ldots V_2 \sigma(V_1 x) \ldots)$. We train this normalized network with SGD on the square loss with Lagrange multipliers and Weight Decay. This architecture differs from the one considered in section 3.4 in that it has C outputs instead of a scalar output. Let the output of the network be $\rho f_V(x) = [\rho f_V^{(1)}(x) \ldots \rho f_V^{(C)}(x)]^\top$, and the target vectors be $y_n = [y_n^{(1)} \ldots y_n^{(C)}]^\top$. We will also follow the notation of (12) and use $h : \mathbb{R}^d \to \mathbb{R}^p$ to denote the last layer features of the deep network. This means that $f_V^{(c)}(x) = \langle V_L^c, h(x) \rangle$. The squared loss function with weight decay is written as $\mathcal{L}_{\mathcal{S}}(\rho, \{V_k\}_{k=1}^{L}) = \frac{1}{NC} \sum_{c=1}^{C} \sum_{n=1}^{N} ||y_{cn} - \rho f_V(x_{cn})||^2 + \lambda \rho^2$.

[Symmetric Quasi-interpolation (Multiclass Classification)] Consider a C-class classification problem with inputs in a feature space $\mathcal{X}$ and labels space $\mathbb{R}^C$. A classifier $f : \mathcal{X} \to \mathbb{R}^C$ symmetrically quasi-interpolates a training dataset $\mathcal{S} = \cup_{c=1}^{C} \mathcal{S}_c = \cup_{c=1}^{C} \{(x_{cn}, y_{cn})\}_{n=1}^{N}$ if for all training examples x_{cn}, $f(x_{cn}) \propto \tilde{e}_c$.
Similar to the binary classification case, we show that this property arises from an analysis of the squared loss landscape for multiclass classification.

Lemma 8 *An overparameterized deep ReLU classifier trained to convergence under the squared loss in the presence of weight decay and weight normalization (WN) satisfies the symmetric quasi-interpolation property*

Proof
Consider the regularized square loss $\mathcal{L}_{\mathcal{S}} = \frac{1}{CN} \sum_{c=1}^{C} \sum_{n=1}^{N} \|\rho f_V(x_{cn}) - \tilde{e}_c\|^2 + \lambda \rho^2$. In the multiclass case we define the first order statistics of the output of the normalized network as $\mu = \frac{1}{CN} \sum_{c=1}^{C} \sum_{n=1}^{N} \langle f_V(x_{cn}), \tilde{e}_c \rangle$, and $M = \frac{1}{CN} \sum_{c=1}^{C} \sum_{n=1}^{N} \|f_V(x_{cn})\|^2$. We consider deep networks that are overparameterized enough to attain 100% accuracy on the training dataset, which means $\langle f_V(x_{cn}), \tilde{e}_c \rangle > 0$. Since the weights of the deep network $\{V_k\}_{k=1}^{L}$ are normalized and the data x_{cn} lie within the unit norm ball, we also have that $\|f_V(x_{cn})\| \leq 1$. However, similar to the binary case, we observe that the norm of $f_V(x_{cn})$ takes values of the order of 10^{-3}.
Using these definitions, we can rewrite the deep network training problem as:

$$\min_{\rho, \{V_k\}_{k=1}^{L}} \mathcal{L}_{\mathcal{S}} = (M + \lambda)\rho^2 - 2\rho\mu + \frac{C}{C-1} \tag{36}$$

All critical points (including minima) need to satisfy $\frac{\partial \mathcal{L}_{\mathcal{S}}}{\partial \rho} = 0$, from which we get $\rho = \frac{\mu}{M+\lambda}$. If we plug this back into the loss, our minimization problem becomes:

$$\begin{aligned} &\min_{\{V_k\}_{k=1}^{L}} (M + \lambda) \times \left(\frac{\mu}{M+\lambda} \right)^2 - 2\frac{\mu^2}{M+\lambda} + \frac{C}{C-1} \\ &= \min_{\{V_k\}_{k=1}^{L}} \frac{C}{C-1} - \frac{\mu^2}{M+\lambda} \end{aligned} \tag{37}$$

Hence in order to minimize the loss we have to find $\{V_k\}_{k=1}^{L}$ that maximizes $\frac{\mu^2}{M+\lambda}$. Since the network is expressive enough to attain any value, and the norm of $f_V(x_{cn})$ is bounded, we see that the loss is minimized when μ^2 is maximized. That is, when $f(x_{cn}) \propto \tilde{e}_c$ for all training examples. ■

We now consider the optimization of the squared loss on deep networks with Weight Normalization and Weight Decay:

$$\mathcal{L}_{\mathcal{S}}(\rho, \{V_k\}_{k=1}^{L}) = \frac{1}{NC} \sum_{c=1}^{C} \sum_{n=1}^{N} ||y_{cn} - \rho f_V(x_{cn})||^2 + \sum_{k=1}^{L} \nu_k \left(||V_k||^2 - 1 \right) + \lambda \rho^2 \tag{38}$$

At each time point t the optimization process selects a random class-balanced batch $\mathcal{S}' = \cup_{c=1}^{C} \cup_{n=1}^{b} \mathcal{S}'_c$ including B samples per-class from $\mathcal{S}'_c \subset \mathcal{S}_c$ and updates the scale and weights of the network is the following manner $V \leftarrow V - \eta \frac{\partial \mathcal{L}_{\mathcal{S}'}(\rho, V)}{\partial V}$, $\rho \leftarrow \rho - \eta \frac{\partial \mathcal{L}_{\mathcal{S}'}(\rho, V)}{\partial \rho}$ where $\eta > 0$ is a predefined learning rate and b is a divisor of N. A convergence point of the optimization process is a point (ρ, V) that will never be updated by any possible sequence of steps taken by the optimization algorithm. Specifically, the convergence points of the proposed method are all points ρ, V for which $\frac{\partial \mathcal{L}_{\mathcal{S}'}(\rho, V)}{\partial V} = 0$ and $\frac{\partial \mathcal{L}_{\mathcal{S}'}(\rho, V)}{\partial \rho} = 0$ for all class-balanced batches $\mathcal{S}' \subset \mathcal{S}$.

Theorem 8 *Let $\mathcal{S} = \cup_{c=1}^{C} \{(x_{cn}, c)\}_{n=1}^{N}$ be a dataset and B be a divisor of N. Let (ρ, V) be the parameters of a ReLU network f_W such that V_L has converged when training using SGD with balanced batches of size $B = bC$ on the square loss with LN+WD. Let $\mu_c = \frac{1}{N} \sum_{n=1}^{N} h(x_{cn})$, $\mu_G = \frac{1}{C} \sum_{c=1}^{C} \mu_c$ and $M = [\dots \mu_c - \mu_G \dots] \in \mathbb{R}^{p \times C}$. Consider points of convergence of SGD that satisfy Property 5.2. Then, those points also satisfy the conditions of Neural Collapse as described below.*

- *NC1: $\mu_c = h(x_{cn})$ for all $n \in [N]$;*

- *NC2: the vectors* $\{\mu_c - \mu_G\}_{c=1}^C$ *form an ETF;*
- *NC3:* $V_L^\top = \frac{M}{\|M\|_F}$*;*
- *NC4:* $\arg\max_{c\in[C]} f_W^c(x) = \arg\min_{c\in[C]} \|\mu_c - h(x)\|$.

Proof Our training objective is the loss function described in (38). The network is trained using SGD along with Lagrange normalization and weight decay. We use SGD with balanced batches to train the network. Each step taken by SGD takes the form $-\eta\frac{\partial \mathcal{L}_{\mathcal{S}'}}{\partial V}$, where $\mathcal{S}' \subset \mathcal{S}$ is a balanced batch containing exactly b samples per class. We consider limit points of the learning procedure, meaning that $\frac{\partial \mathcal{L}_{\mathcal{S}'}}{\partial V} = 0$ for all balanced batches $\mathcal{S}'$. Let $\mathcal{S}' = \cup_{c=1}^C \cup_{n=1}^b \{(\hat{x}_{cn}, \hat{y}_{cn})\}$ be such a balanced batch. We use SGD, where at each time t the batch $\mathcal{S}'$ is drawn at random from $\mathcal{S}$, to study the time evolution of the normalized parameters V_L in the limit $\eta \to 0$.

$$\begin{aligned} V_L^{(t+1)} &= V_L^{(t)} - \eta\frac{\partial \mathcal{L}_{\mathcal{S}'}}{\partial V_L^{(t)}} \\ \implies V_L^{(t+1)} &= V_L^{(t)} - \eta \times \left(\frac{1}{B}\sum_{c'=1}^{C}\sum_{n=1}^{b} 2\rho(\rho f_V(x_{c'n}) - \tilde{e}_{c'})h(x_{c'n})^\top + 2\nu_L^{(t)}V_L^{(t)}\right) \end{aligned} \tag{39}$$

We can apply the unit norm constraints $||V_L^{(t+1)}||_F^2 = \text{tr}(V_L^{(t+1)\top}V_L^{(t+1)}) = 1$ and $||V_L^{(t)}||_F^2 = \text{tr}(V_L^{(t)\top}V_L^{(t)}) = 1$ and ignore all terms that are $O\left(\eta^2\right)$ to compute $\nu_L^{(t)}$ as:

$$\begin{aligned} 2\nu_L^{(t)} &= -\frac{1}{B}\sum_{c'=1}^{C}\sum_{n=1}^{b} 2\rho\text{tr}\left(V_L^{(t)\top}(\rho f_V(x_{c'n}) - \tilde{e}_{c'})h(x_{c'n})^\top\right) \\ \implies \nu_L^{(t)} &= -\frac{1}{B}\sum_{c'=1}^{C}\sum_{n=1}^{b} \rho\text{tr}\left((V_L^{(t)}h(x_{c'n}))^\top(\rho f_V(x_{c'n}) - \tilde{e}_{c'})\right) \\ &= -\frac{1}{B}\sum_{c'=1}^{C}\sum_{n=1}^{b} \rho f_V(x_{c'n})^\top(\rho f_V(x_{c'n}) - \tilde{e}_{c'}) \end{aligned} \tag{40}$$

This means that the (stochastic) gradient of the loss with respect to the last layer V_L, and each classifier vector V_L^c with Lagrange Normalization can be written as (we drop the time index t for clarity):

$$\begin{aligned} \frac{\partial \mathcal{L}_{\mathcal{S}'}}{\partial V_L} &= \frac{-2\rho}{B}\sum_{c'=1}^{C}\sum_{n=1}^{b} f_V(x_{c'n})^\top(\rho f_V(x_{c'n}) - \tilde{e}_{c'})V_L - (\rho f_V(x_{c'n}) - \tilde{e}_{c'})h(x_{c'n})^\top \\ \frac{\partial \mathcal{L}_{\mathcal{S}'}}{\partial V_L^c} &= \frac{-2\rho}{B}\sum_{c'=1}^{C}\sum_{n=1}^{b} f_V(x_{c'n})^\top(\rho f_V(x_{c'n}) - \tilde{e}_{c'})V_L^c - (\rho f_V^{(c)}(x_{c'n}) - \tilde{e}_{c'}^{(c)})h(x_{c'n}) \end{aligned} \tag{41}$$

Let us analyze the equilibrium parameters at the last layer, considering each classifier vector V_L^c of V_L, separately:

$$\begin{aligned} 0 = \frac{\partial \mathcal{L}_{\mathcal{S}'}}{\partial V_L^c} &= \frac{-2\rho}{B}\sum_{c'=1}^{C}\sum_{n=1}^{b} f_V(x_{c'n})^\top(\rho f_V(x_{c'n}) - \tilde{e}_{c'})V_L^c - (\rho f_V^{(c)}(x_{c'n}) - \tilde{e}_{c'}^{(c)})h(x_{c'n}) \\ &= \frac{-2\rho}{B}\sum_{n=1}^{b} f_V(x_{cn})^\top(\rho f_V(x_{cn}) - \tilde{e}_c)V_L^c - (\rho f_V^{(c)}(x_{cn}) - 1)h(x_{cn}) \\ &\quad - \frac{2\rho}{B}\sum_{c'\in[C]\setminus\{c\}}\sum_{n=1}^{b} f_V(x_{c'n})^\top(\rho f_V(x_{c'n}) - \tilde{e}_{c'})V_L^c - \left(\rho f_V^{(c)}(x_{c'n}) + \frac{1}{C-1}\right)h(x_{c'n}) \end{aligned} \tag{42}$$

Using Property 5.2 and considering solutions that achieve *symmetric quasi-interpolation*, with $\rho f_V(\hat{x}_{cn}) = \alpha\tilde{e}_c$, we have

$$\frac{2\rho}{B}\sum_{n=1}^{b}(\alpha-1)h(x_{cn})-\frac{2\rho}{B}\sum_{c'\in[C]\setminus\{c\}}\sum_{n=1}^{b}\frac{\alpha-1}{C-1}h(x_{c'n})-\frac{2\alpha(\alpha-1)C}{C-1}V_L^c=0. \tag{43}$$

In addition, consider a second batch $\mathcal{S}''$ that differs from $\mathcal{S}'$ by only one sample x'_{cn} instead of x_{cn} from class c. By applying the previous Eq. (43) for $\mathcal{S}'$ and for $\mathcal{S}''$, we can obtain $h(x_{cn})=h(x'_{cn})$, which proves NC1.

Let $\mathcal{S}=\cup_{i=1}^{k}\mathcal{S}^i$ be a partition of $\mathcal{S}$ into $k=N/b$ (an integer) disjoint batches. Since our data is balanced, we obtain that

$$\begin{aligned}
0&=\frac{1}{k}\sum_{i=1}^{k}\frac{\partial\mathcal{L}_{\mathcal{S}^i}(\rho,V)}{\partial V_L^c}\\
&=\frac{\partial\mathcal{L}_{\mathcal{S}}(\rho,V)}{\partial V_L^c}\\
&=\frac{2\rho}{NC}\sum_{c'=1}^{C}\sum_{n=1}^{N}f_V(x_{c'n})^\top(\rho f_V(x_{c'n})-\tilde{e}_{c'})V_L^c-(\rho f_V^{(c)}(x_{c'n})-\tilde{e}_{c'}^{(c)})h(x_{c'n})\\
&=\frac{2\rho}{NC}\sum_{n=1}^{N}(\alpha-1)h(x_{cn})-\frac{2\rho}{NC}\sum_{c'\in[C]\setminus\{c\}}\sum_{n=1}^{N}\frac{\alpha-1}{C-1}h(x_{c'n})-\frac{2\alpha(\alpha-1)C}{C-1}V_L^c
\end{aligned} \tag{44}$$

Under the conditions of NC1 we can simply write $\mu_c=h(x_{cn})$ for all $n\in[N]$ and $c\in[C]$. Let us denote the global feature mean by $\mu_G=\frac{1}{C}\sum_{c=1}^{C}\mu_c$. This means we have:

$$\frac{\partial\mathcal{L}_{\mathcal{S}}(\rho,V)}{\partial V_L^c}=0\quad\implies V_L^c=\frac{\rho}{\alpha C}\cdot(\mu_c-\mu_G). \tag{45}$$

This implies that the last layer parameters V_L are a scaled version of the centered class-wise feature matrix $M=[\dots\mu_c-\mu_G\dots]$. Thus at equilibrium, with quasi interpolation of the training labels, we obtain $\frac{V_L^\top}{\|V_L\|_F}=\frac{M}{\|M|_F}$.

From the SGD equations, we can also see that at equilibrium, with quasi interpolation, all classifier vectors in the last layer (V_L^c, and hence $\mu_c-\mu_G$) have the same norm:

$$\begin{aligned}
\|V_L^c\|_2^2&=\frac{\frac{1}{NC}\sum_{c'=1}^{C}\sum_{n=1}^{N}(\rho f_V^{(c)}(x_{c'n})-\tilde{e}_{c'}^{(c)})\rho f_V^{(c)}(x_{c'n})}{\frac{1}{NC}\sum_{c'=1}^{C}\sum_{n=1}^{N}\langle\rho f_V(x_{c'n})-\tilde{e}_{c'},\rho f_V(x_{c'n})\rangle}\\
&=\frac{\frac{\alpha(\alpha-1)}{C}+\frac{\alpha(\alpha-1)}{C(C-1)}}{\alpha(\alpha-1)\times\frac{C}{C-1}}=\frac{1}{C}
\end{aligned} \tag{46}$$

From the quasi-interpolation of the correct class label we have that $\langle V_L^c,\mu_c\rangle=\frac{\alpha}{\rho}$ which means $\langle V_L^c,\mu_G\rangle+\langle V_L^c,\mu_c-\mu_G\rangle=\frac{\alpha}{\rho}$. Now using Equation (45)

$$\begin{aligned}
\langle V_L^c,\mu_G\rangle&=\frac{\alpha}{\rho}-\frac{\alpha C}{\rho}\|V_L^c\|_2^2\\
&=\frac{\alpha}{\rho}-\frac{\alpha C}{\rho}\times\frac{1}{C}=0
\end{aligned} \tag{47}$$

From the quasi-interpolation of the incorrect class labels, we have that $\langle V_L^c,\mu_{c'}\rangle=\frac{-\alpha}{\rho(C-1)}$, which means $\langle V_L^c,\mu_{c'}-\mu_G\rangle+\langle V_L^c,\mu_G\rangle=\frac{-\alpha}{\rho(C-1)}$. Plugging in the previous result and using (46) yields

$$\begin{aligned}
&\frac{\alpha C}{\rho}\times\langle V_L^c,V_L^{c'}\rangle=\frac{-\alpha}{\rho(C-1)}\\
&\implies\langle\tilde{V}_L^c,\tilde{V}_L^{c'}\rangle=\frac{1}{\|V_L^c\|_2^2}\times\frac{-1}{C(C-1)}=-\frac{1}{C-1}.
\end{aligned} \tag{48}$$

Here $\tilde{V}_L^c=\frac{V_L^c}{\|V_L^c\|_2}$, and we use the fact that all the norms $\|V_L^c\|_2$ are equal. This completes the proof that the normalized classifier parameters form an ETF. Moreover since $V_L^c\propto\mu_c-\mu_G$ and all the proportionality constants are independent of c, we obtain $\sum_c V_L^c=0$. This completes the proof of the NC2 condition. NC4 follows then from NC1-NC2, as shown by theorems in (12). ■

Remarks

- The analysis of the loss landscape and of the qualitative dynamics under the square loss in section 3.5 and in section 3.3 implies that all quasi-interpolating solutions with $\rho \geq \rho_0$ and $\lambda > 0$ that satisfying assumption 5.2 yield Neural Collapse and have its four properties.
- SGD is a necessary requirement in our proof of NC1.
- Our analysis implies that there is no direct relation between Neural Collapse and generalization. In fact, a careful look at our derivation suggests that NC1 to NC4 should take place for any quasi-interpolating solutions (in the square loss case), including solutions that do not have a large margin. In particular, our analysis predicts Neural Collapse for datasets with fully random labels – a prediction which has been experimentally verified.

6 SGD bias towards low-rank weight matrices and intrinsic SGD noise

In the previous sections we assumed that ρ and V_k are trained using GF. In this section we consider a slightly different setting where SGD is applied instead of GF. Specifically, V_k and ρ are first initialized and then iteratively updated simultaneously in the following manner

$$\begin{aligned} \rho &\leftarrow \rho - \eta \frac{\partial \mathcal{L}_{\mathcal{S}'}(\rho, \{V_k\}_{k=1}^L)}{\partial \rho} = \rho - \eta \frac{2}{B} \sum_{(x_n, y_n) \in \mathcal{S}'} (1 - \rho \bar{f}_n) \bar{f}_n - 2\eta\lambda\rho \\ V_k &\leftarrow V_k - \frac{\partial \mathcal{L}_{\mathcal{S}'}(\rho, \{V_k\}_{k=1}^L)}{\partial V_k} = V_k - \eta \frac{2}{B} \sum_{(x_n, y_n) \in \mathcal{S}'} (1 - \rho \bar{f}_n) \rho \frac{\partial \bar{f}_n}{\partial V_k} - 2\eta\nu_k V_k. \end{aligned} \tag{49}$$

where $\mathcal{S}'$ is selected uniformly as a subset of $\mathcal{S}$ of size B, $\eta > 0$ is the learning rate and ν_k is computed according to (4) with $\mathcal{S}$ replaced by $\mathcal{S}'$.

6.1 Low-rank bias

An intriguing argument for small rank weight matrices is the following observation that follows from Equation (5) (see also (7)).

Lemma 9 *Let f_W be a neural network. Assume that we iteratively train ρ and $\{V_k\}_{k=1}^L$ using the process described above with weight decay $\lambda > 0$. Suppose that training converges, that is $\frac{\partial \mathcal{L}_{\mathcal{S}'}(\rho, \{V_k\}_{k=1}^L)}{\partial \rho} = 0$ and $\forall\, k \in [L]:\ \frac{\partial \mathcal{L}_{\mathcal{S}'}(\rho, \{V_k\}_{k=1}^L)}{\partial V_k} = 0$ for all mini-batches $\mathcal{S}' \subset \mathcal{S}$ of size $B < |\mathcal{S}|$. Assume that $\forall\, n \in [N]:\ \bar{f}_n \neq 0$. Then, the ranks of the matrices V_k are at most ≤ 2.*

Proof Let $f_V(x) = V_L \sigma(V_{L-1} \dots \sigma(V_1 x) \dots)$ be the normalized neural network, where $V_l \in \mathbb{R}^{d_{l+1} \times d_l}$ and $\|V_l\| = 1$ for all $l \in [L]$. We would like to show that the matrix $\frac{\partial f_V(x)}{\partial V_k}$ is of rank ≤ 1. We note that for any given vector $z \in \mathbb{R}^d$, we have $\sigma(v) = \text{diag}(\sigma'(v)) \cdot v$ (where σ is the ReLU activation function). Therefore, for any input vector $x \in \mathbb{R}^n$, the output of f_V can be written as follows,

$$\begin{aligned} f_V(x) &= V_L \sigma(V_{L-1} \dots \sigma(V_1 x) \dots) \\ &= V_L \cdot D_{L-1}(x; V) \cdots D_1(x; V) \cdot V_1 \cdot x, \end{aligned} \tag{50}$$

where $D_l(x; V) = \text{diag}[\sigma'(u_l(x; V)))]$ and $u_l(x; V) = V_l \sigma(V_{l-1} \dots \sigma(V_1 x) \dots)$. We denote by $u_{l,i}(x; V)$ the i'th coordinate of the vector $u_l(x; V)$. We note that $u_l(x; V)$ are continuous functions of V. Therefore, assuming that none of the coordinates $u_{l,i}(x; V)$ are zero, there exists a sufficiently small ball around V for which $u_{l,i}(x; V)$ does not change its sign. Hence, within this ball, $\sigma'(u_{l,i}(x; V))$ are constant. We define a set $\mathcal{V} := \{V \mid \forall l \leq L:\ \|V_l\| = 1\}$ and $\mathcal{V}_{l,i} = \{V \in \mathcal{V} \mid u_{l,i}(x; V) = 0\}$. We note that as long as $x \neq 0$, the set $\mathcal{V}_{l,i}$ is negligible within $\mathcal{V}$. Since there is a finite set of indices l, i, the set $\bigcup_{l,i} \mathcal{V}_{l,i}$ is also negligible within $\mathcal{V}$.
Let V be a set of matrices for which none of the coordinates $u_{l,i}(x; V)$ are zero. Then, the matrices $\{D_l(x; V)\}_{l=1}^{L-1}$ are constant in the neighborhood of V, and therefore, their derivative with respect to

V_k are zero. Let $a^\top = V_L \cdot D_{L-1}(x;V)V_{L-1}\cdots V_{k+1}D_k(x;V)$ and $b = D_{k-1}(x)\cdot V_{k-1}\cdots V_1 x$. We can write $f_V(x) = a(x;V)^\top \cdot V_k \cdot b(x;V)$. Since the derivatives of $a(x;V)$ and $b(x;V)$ with respect to V_k are zero, by applying $\frac{\partial a^\top X b}{X} = ab^\top$, we have $\frac{\partial f_V(x)}{\partial V_k} = a(x;V)\cdot b(x;V)^\top$ which is a matrix of rank at most 1. Therefore, $\frac{\partial \bar{f}_n}{\partial V_k} = y_n \frac{\partial f_V(x_n)}{\partial V_k}$ is a matrix of rank at most 1. Therefore, for any input $x_n \neq 0$, with measure 1, $\frac{\partial \bar{f}_n}{\partial V_k}$ is a matrix of rank at most 1.

Since $\forall\, k \in [L]:\ \frac{\partial \mathcal{L}_{\mathcal{S}'}(\rho,\{V_k\}_{k=1}^L)}{\partial V_k} = 0$ for all mini-batches $\mathcal{S}' = \{(x_{i_j}, y_{i_j})\}_{j=1}^B \subset \mathcal{S}$ of size $B < |\mathcal{S}|$, we have

$$\frac{\partial \mathcal{L}_{\mathcal{S}'}(\rho,\{V_k\}_{k=1}^L)}{\partial V_k} = \frac{2}{B}\rho \sum_{j=1}^{B}\left[\left(1-\rho\bar{f}_{i_j}\right)\left(-V_k\bar{f}_{i_j} + \frac{\partial \bar{f}_{i_j}}{\partial V_k}\right)\right] = 0. \tag{51}$$

Since interpolation is impossible when training with $\lambda > 0$, there exists at least one $n \in [N]$ for which $\rho\bar{f}_n \neq 1$. We consider two batches $\mathcal{S}_i'$ and $\mathcal{S}_j'$ of size B that differ by sample, (x_i, y_i) and (x_j, y_j). We have

$$\begin{aligned}\forall\, i,j \in [N]:\ 0 &= \frac{\partial \mathcal{L}_{\mathcal{S}_i'}(\rho,\{V_k\}_{k=1}^L)}{\partial V_k} - \frac{\partial \mathcal{L}_{\mathcal{S}_j'}(\rho,\{V_k\}_{k=1}^L)}{\partial V_k} \\ &= \frac{2}{B}\cdot\rho\left[\left(1-\rho\bar{f}_i\right)\left(-V_k\bar{f}_i + \frac{\partial \bar{f}_i}{\partial V_k}\right) - \left(1-\rho\bar{f}_j\right)\left(-V_k\bar{f}_j + \frac{\partial \bar{f}_j}{\partial V_k}\right)\right].\end{aligned} \tag{52}$$

Assume that there exists a pair $i,j \in [N]$ for which $(1-\rho\bar{f}_i)\bar{f}_i \neq (1-\rho\bar{f}_j)\bar{f}_j$. Then, we can write

$$V_k = \frac{\left[(1-\rho\bar{f}_i)\cdot\frac{\partial \bar{f}_i}{\partial V_k} + (1-\rho\bar{f}_j)\cdot\frac{\partial \bar{f}_j}{\partial V_k}\right]}{[(1-\rho\bar{f}_i)\bar{f}_i - (1-\rho\bar{f}_j)\bar{f}_j]}. \tag{53}$$

Since $\frac{\partial \bar{f}_i}{\partial V_k}$ and $\frac{\partial \bar{f}_j}{\partial V_k}$ are matrices of rank ≤ 1 (see the analysis above), we obtain that V_k is of rank ≤ 2. Otherwise, assume that for all pairs $i,j \in [N]$, we have $\alpha = (1-\rho\bar{f}_i)\bar{f}_i = (1-\rho\bar{f}_j)\bar{f}_j$. In this case we obtain that for all $i,j \in [N]$, we have

$$\left(1-\rho\bar{f}_i\right)\cdot\frac{\partial \bar{f}_i}{\partial V_k} = \left(1-\rho\bar{f}_j\right)\cdot\frac{\partial \bar{f}_j}{\partial V_k} = U. \tag{54}$$

Therefore, since $\alpha = (1-\rho\bar{f}_i)\bar{f}_i = (1-\rho\bar{f}_j)\bar{f}_j$, by Equation 51,

$$0 = \frac{2}{B}\rho\sum_{j=1}^{B}\left[\left(1-\rho\bar{f}_{i_j}\right)\left(-V_k\bar{f}_{i_j} + \frac{\partial \bar{f}_{i_j}}{\partial V_k}\right)\right] = -2\rho\alpha V_k + 2\rho U. \tag{55}$$

Since the network cannot perfectly fit the dataset when trained with $\lambda > 0$, we obtain that there exists $i \in [N]$ for which $(1-\rho\bar{f}_i) \neq 0$. Since $\bar{f}_i \neq 0$ for all $i \in [N]$, this implies that $\alpha \neq 0$. We conclude that V_k is proportional to U which is of rank ≤ 1.

■

6.1.1 Is Low-Rank Bias Related to Generalization?

An obvious question is whether a deep ReLU network that fits the data generalizes better than another one if the rank of its weight matrices is lower. A forthcoming paper (53) proves that

Theorem 9 (*informal*) *Let f_W be a normalized neural network, trained with SGD under square loss in the presence of WN. Assume that the weight matrix W_k of dimensionality n, n has rank $r < n$. Then its contribution to the Rademacher complexity of the network will be $\sqrt{\frac{r}{n}}$ instead of just 1 as in the typical bound.*

Proof [Sketch proof] We start assuming W^k of norm 1 that is $\|W^k\| = \|V^k\| = 1$ and we consider the following quantity that appears in the peeling process of deriving the Rademacher complexity of a network:

$$\mathbb{E}_\epsilon \sup_{W^k} \frac{1}{m}\sqrt{\|\sum_{i=1}^{m}\epsilon_i W^k\sigma(x_i)\|^2}. \tag{56}$$

then assume that W^k of dimensionality $n \times n$ becomes at convergence of rank r whereas the input activities $h^i_{k-1} = z_i$ has norm $\|z_i\| = 1$ and dimensionality n. We also assume that in expectation z_i have components of uniform norm in terms of the standard basis in $\mathbb{R}^n$. We rewrite

$$\mathcal{R}_k = \mathbb{E}_\epsilon \sup_{\Sigma,U,V^T} \frac{1}{m} \sqrt{\| \sum_{i=1}^m \epsilon_i U \Sigma V^T z_i \|^2}. \tag{57}$$

Since V^T is defined to be orthonormal $z' = V^T z$ has the same norm of z. Thus

$$\begin{aligned} \mathcal{R}_k = \mathbb{E}_\epsilon \sup_{\Sigma} \frac{1}{m} \sqrt{\| \sum_{i=1}^m \epsilon_i U \Sigma z' \|^2} &= \mathbb{E}_\epsilon \sup_{\Sigma} \frac{1}{m} \sqrt{\| \sum_{i=1}^m \epsilon_i U z^{(r)} \|^2} \\ &= \mathbb{E}_\epsilon \frac{1}{m} \sqrt{\|U\|^2 \|z^{(r)}\|^2 \| \sum_{i=1}^m \epsilon_i \|^2} \\ &= \sqrt{\frac{2}{m}} \|z^{(r)}\| = \sqrt{\frac{2}{m}} \sqrt{\frac{r}{n}}, \end{aligned} \tag{58}$$

where $z^{(r)}$ consists of the vector with r components of z'. ■

6.2 Origin of SGD noise

Lemma 9 shows that there cannot be convergence to a unique set of weights $\{V_k\}_{k=1}^L$ that satisfy equilibrium for all minibatches. More details of the argument are illustrated in (54; 55). When $\lambda = 0$, interpolation of all data points is expected: in this case, the GD equilibrium can be reached without any constraint on the weights. This is also the situation in which SGD noise is expected to essentially disappear: compare the histograms on the left and the right hand side of Figure 13. Thus, during training, the solution $\{V_k\}_{k=1}^L$ is not the same for all samples: there is *no convergence to a unique solution* but instead fluctuations between solutions during training. The absence of convergence to a unique solution is not surprising for SGD when the landscape is not convex.

7 Summary

The dynamics of GF In this paper we have considered a model of the dynamics of, first, gradient flow, and then Stochastic Gradient Descent, in overparametrized ReLU neural networks trained for square loss minimization. Under the assumption of convergence to zero loss minima, we have shown that solutions have a bias toward small ρ, defined as the product of the Frobenius norms of each layer's (unnormalized) weight matrix. We assume that during training there is normalization using a Lagrange multiplier (LM) of each layer weight matrix but the last one, together with Weight Decay (WD) with the regularization parameter λ. Without weight decay, the best solution would be the interpolating solution with minimum ρ that may be achieved with appropriate initial conditions are appropriate.

Remarks

- The bias towards small ρ solutions induced by regularization with $\lambda > 0$ may be replaced – when $\lambda = 0$ – by an implicit bias induced by small initialization. With appropriate parameter values, small initialization allows convergence to the first quasi-interpolating solution for increasing ρ from ≈ 0 to ρ_0. For $\lambda = 0$ we have empirically observed solutions with large ρ that are suboptimal and probably similar to the NTK regime.
- A puzzle that remains open is why BN leads to better solutions than LN and WN, despite similarities between them. WN is easier to formalize mathematically as LN, which is the main reason for the role it plays in this paper.

Generalization and bounds Building on our analysis of the dynamics of ρ we derive new norm-based generalization bounds for CNNs for the special case of non-overlapping convolutional patches. These bounds show a) that generalization for CNNs can be orders of magnitude better than for dense networks and b) that these bounds can be empirically loose but non-vacuous despite overparametrization.

Remarks

- For $\lambda > 0$ a main property of the minimizers that upper bounds their expected error is ρ, which is the inverse of the margin: we prove that among all the quasi-interpolating solutions the ones associated with smaller ρ have better bounds on the expected classification error.
- The situation here is somewhat similar to the linear case: for overparametrized networks the best solution in terms of generalization is the minimum norm solution towards which GD is biased.
- Large margin is usually associated with good generalization (56); in the meantime, however, it is also broadly recognized that margin alone does not fully account for generalization in deep nets (28; 57; 31). Margin in fact provides an upper bound on generalization error, as shown in section 4. Larger margin gives a better upper bound on the generalization error for the same network trained on the same data. We have verified empirically this property by varying the margin using different degrees of random labels in a binary classification task. While training gives perfect classification and zero square loss, the margin on the training set together with the test error decreases with the increase in the percentage of random labels. Of course large margin in our theoretical analysis is associated with regularization which helps minimizing ρ. Since ρ is the product of the Frobenius norm, its minimization is directly related to minimizing a Bayes prior(58) which is itself directly related to minimum description length principles.
- We do not believe that flat minima directly affect generalization. As we described in an earlier section, degenerate minima correspond to solutions that have zero empirical loss (for $\lambda = 0$). Minimizing the empirical loss is a (almost) necessary condition for good generalization. It is not, however, sufficient since minimization of the expected error also requires a solution with low complexity.
- The upper bound given in section 4, however, does not explain by itself details of the generalization behavior that we observe for different initializations (see Figure 6), where small differences in margin are actually anticorrelated with small differences in test error. We conjecture that margin (related to ρ) together with rank (related to $\mathbb{R}_N(\mathbb{F})$) may be sufficient to explain generalization.

Neural Collapse Another consequence of our analysis is a proof of Neural Collapse for deep networks trained with square loss in the binary classification case without any assumption. In particular, we prove that training the network using SGD with weight decay, induces a bias towards low-rank weight matrices and yields SGD noise in the weight matrices and in the margins, which makes exact convergence impossible, even asymptotically.

Remarks

- A natural question is whether Neural Collapse is related to solutions with good generalization. Our analysis suggests that this is not the case, at least not directly: Neural Collapse is a property of the dynamics, independently of the size of the margin which provides an upper bound on the expected error. In fact, our prediction of Neural Collapse for randomly labeled CIFAR10, was confirmed originally in then preliminary experiments by our collaborators (Papyan et al.) and more recently in other papers (see for instance, (33)).
- Margins, however, do converge to each other but only within a small ϵ, implying that the first condition for Neural Collapse (12) is satisfied only in this approximate sense. This is equivalent to saying that that SGD does not converge to a unique solution that corresponds to zero gradient for all data point.

Intertwining of SGD noise, rank minimization, minimum ρ, Neural Collapse, Stiefel manifolds
Consider the equations

$$\dot{\rho} = -\frac{2}{B}[\sum_{n\in B} \rho(f_n)^2 - \sum_{n\in B} f_n y_n] - 2\lambda\rho \tag{59}$$

and

$$\dot{V_k} = \frac{2\rho}{B} \sum_{n\in B} [(\rho f_n - y_n)(V_k f_n - \frac{\partial f_n}{\partial V_k})] \tag{60}$$

At convergence there is quasi-interpolation $(\rho f_n - y_n) \leq \epsilon > 0$ implying $\dot{V_k}$ small at all layers, if the term $V_k f_n - \frac{\partial f_n}{\partial V_k}$ is not large (since $\|\dot{V_k}\| \leq \frac{2\rho\epsilon}{B^2} \| \sum_{n\in B}(V_k f_n - \frac{\partial f_n}{\partial V_k})\|$).
Recall now that if $\| \sum_{n\in B}(V_k f_n - \frac{\partial f_n}{\partial V_k})\|$ is small for $k = k^*$ then

$$V_{k^*} f = [V_L D_{L-1}(x) V_{L-1} \cdots V_{k+1} D_k(x)]^T \, D_{k-1}(x) V_{k-1} D_{k-2}(x) \cdots D_1(x) V_1 x = ab^T \tag{61}$$

because $f(x) = a^T V_k b$ and $\frac{\partial f}{\partial V_k} = ab^T$.
Suppose $f(x) = V_L D V_{k^*+1} D V_{k^*} z$ with the matrices being orthogonal. Then it is easy to check that the constraint equations yield $V_3 \propto \frac{\partial f}{\partial V_3} = V_4^T (V_2 V_1)^T$ and $V_2 \propto (V_4 V_3)^T (V_1)^T$. Together they satisfy $V_3 = V_3$. These constraints equations are satisfied if the V_k are Stiefel matrices (since then Because of this property the constraint equations are always satisfied. The underlying reason for restricting this class of solutions to the orthogonal group is WN, since they are equivalent to constrained optimization with Lagrange multipliers. Regularization of each weight matrix of a linear network reduces the symmetry group of the loss function from the general linear group to the orthogonal group . Furthermore, orthogonal matrices are the inverse matrices of minimum total squared Frobenius norm (sum of the squared singular values). In general we should consider non-square matrices in an orthogonal Stiefel manifold on the sphere. The Stiefel manifold can be thought of as a set of matrices by writing a k-frame as a matrix of k column vectors. The orthonormality condition is expressed by , where denotes the k × k identity matrix. Every orthogonal transformation of a k-frame in results in another k-frame, and any two k-frames are related by some orthogonal transformation.

Conclusion Finally, we would like to emphasize that the analysis of this paper supports the idea that the advantage of deep networks relative to other standard classifiers is greater for the problems to which specific deep architectures such as CNNs can be applied. The deep reason is that CNNs reflect the function graph of certain locally compositional target function – which have small intrinsic dimensionality – and thus can be approximated well by sparse networks without incurring in the curse of dimensionality. Despite overparametrization the compositionally sparse networks can then show good generalization.

Acknowledgments

We thank Lorenzo Rosasco, Eran Malach and Shimon Ullman for many relevant discussions. This material is based upon work supported by the Center for Minds, Brains and Machines (CBMM), funded by NSF STC award CCF-1231216. This research was also sponsored by grants from the National Science Foundation (NSF-0640097, NSF-0827427), AFSOR-THRL (FA8650-05-C-7262), BCRIC and Lockheed Martin Space Advanced Technology Center.

References

[1] K. Lyu and J. Li, "Gradient descent maximizes the margin of homogeneous neural networks." *CoRR*, abs/1906.05890, 2019.

[2] T. Poggio, A. Banburski, and Q. Liao, "Theoretical issues in deep networks." *Proceedings of the National Academy of Sciences*, vol. 117, no. 48, pp. 30039-30045, 2020.

[3] M. S. Nacson, S. Gunasekar, J. Lee, N. Srebro, and D. Soudry, "Lexicographic and Depth-Sensitive Margins in Homogeneous and Non-Homogeneous Deep Models". *arXiv e-prints arXiv:1905.07325*, May 2019.

[4] A. Banburski, Q. Liao, B. Miranda et al., "Theory of deep learning III: Dynamics and generalization in deep networks." *Center for Brains, Minds and Machines (CBMM) Memo No. 90*, 2019.

[5] L. Hui and M. Belkin, "Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks." *arXiv preprint arXiv:2006.07322*, 2020.

[6] R.M. Rifkin, "Everything Old Is New Again: A Fresh Look at Historical Approaches to Machine Learning." *PhD thesis, Massachusetts Institute of Technology*, 2002.

[7] T. Poggio and Q. Liao, "Generalization in deep network classifiers trained with the square loss." *Center for Brains, Minds and Machines (CBMM) Memo No. 112*, 2019.

[8] M. Xu, A. Rangamani, A Banburski et al., "Deep Classifiers trained with the Square Loss." *Center for Brains, Minds and Machines (CBMM) Memo No. 117*, 2022.

[9] T. Poggio and Y. Cooper, "Loss landscape: SGD has a better view." *CBMM Memo No. 107*, 2020.

[10] Y. Cooper, "Global minima of overparameterized neural networks." *SIAM Journal on Mathematics of Data Science*, vol. 3, no. 2, pp. 676–691, 2021.

[11] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization." *CoRR*, abs/1611.03530, 2016.

[12] 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*, vol. 117, no. 40, pp.24652–24663, 2020.

[13] N. Timor, G. Vardi, and O. Shamir, "Implicit regularization towards rank minimization in relu networks." *CoRR*, abs/2201.12760, 2022.

[14] D. Soudry, E. Hoffer, M. S. Nacson, S. Gunasekar, and N. Srebro, "The implicit bias of gradient descent on separable data." *The Journal of Machine Learning Research*, vol. 19, n. 1, pp. 2822–2878, 2018.

[15] L. Chizat and F. Bach, "Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss." In *Conference on Learning Theory*, pp. 1305–1338. PMLR, 2020.

[16] T. Xu, Y. Zhou, K. Ji, and Y. Liang, "When will gradient methods converge to max-margin classifier under relu models?" *Stat*, vol. 10, no. 1, pp. e354, 2021.

[17] V. Muthukumar, A. Narang, V. Subramanian et al., "Classification vs regression in overparameterized regimes: Does the loss function matter?" *arXiv e-prints*, page arXiv:2005.08054, May 2020.

[18] T. Liang and A. Rakhlin, "Just Interpolate: Kernel "Ridgeless" Regression Can Generalize." *arXiv e-prints*, page arXiv:1808.00387, Aug 2018.

[19] T. Liang and B. Recht, "Interpolating classifiers make few mistakes." *arXiv preprint arXiv:2101.11815*, 2021.

[20] K. Zhong, Z. Song, P. Jain, P. L. Bartlett, and I. S. Dhillon, "Recovery guarantees for one-hidden-layer neural networks." In *International Conference on Machine Learning*, pp. 4140–4149. PMLR, 2017.

[21] M. Soltanolkotabi, A. Javanmard, and J. D. Lee, "Theoretical insights into the optimization landscape of over-parameterized shallow neural networks." *IEEE Transactions on Information Theory*, vol. 65, no. 2. pp. 742–769, 2018.

[22] S. S. Du, X. Zhai, B. Poczos, and A. Singh, "Gradient descent provably optimizes over-parameterized neural networks." In *International Conference on Learning Representations*, pp. 1–19, 2019.

[23] L. Chizat, E. Oyallon, and F. Bach, "On lazy training in differentiable programming." *arXiv preprint arXiv:1812.07956*, 2018.

[24] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks." *arXiv preprint arXiv:1806.07572*, 2018.

[25] S. Mei, A. Montanari, and P. Nguyen, "A mean field view of the landscape of two-layer neural networks." *Proceedings of the National Academy of Sciences*, vol. 115, no. 33, pp. E7665–E7671, 2018.

[26] Z. Chen, G. M. Rotskoff, J. Bruna, and E. Vanden-Eijnden, "A dynamical central limit theorem for shallow neural networks." *arXiv preprint arXiv:2008.09623*, 2020.

[27] S. Arora, S. Du, W. Hu, Z. Li, and R. Wang, "Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks." In *International Conference on Machine Learning*, pp. 322–332. PMLR, 2019.

[28] P. Bartlett, D. J. Foster, and M. Telgarsky, "Spectrally-normalized margin bounds for neural networks". *Advances in Neural Information Processing Systems* (*NeurIPS*), June 2017.

[29] B. Neyshabur, R. Tomioka, and N. Srebro, "Norm-Based Capacity Control in Neural Networks." In *Conference on Learning Theory*, pp. 1376-1401, PMLR, 2015.

[30] N. Golowich, A. Rakhlin, and O. Shamir, "Size-Independent Sample Complexity of Neural Networks." *arXiv e-prints*, page arXiv:1712.06541, Dec 2017.

[31] Y. Jiang, B. Neyshabur, H. Mobahi, D. Krishnan, and S. Bengio, "Fantastic generalization measures and where to find them." *arXiv preprint arXiv:1912.02178*, 2019.

[32] D. G. Mixon, H. Parshall, and J. Pi, "Neural collapse with unconstrained features." *CoRR*, abs/2011.11619, 2020.

[33] Z. Zhu, T. Ding, J. Zhou et al., "A geometric analysis of neural collapse with unconstrained features." *Advances in Neural Information Processing Systems*, vol. 34, pp. 29820–29834, 2021.

[34] J. Lu and S. Steinerberger, "Neural collapse with cross-entropy loss." *CoRR*, abs/2012.08465, 2020.

[35] C. Fang, H. He, Q. Long, and W. J. Su, "Layer-peeled model: Toward understanding well-trained deep neural networks." *CoRR*, abs/2101.12699, 2021.

[36] W. E and S. Wojtowytsch, "On the emergence of tetrahedral symmetry in the final and penultimate layers of neural network classifiers." *arXiv preprint arXiv:2012.05420*, 2020.

[37] T. Ergen and M. Pilanci, "Revealing the structure of deep neural networks via convex duality" *arXiv preprint arXiv:2002.09773*, 2020.

[38] T. Poggio and Q. Liao, "Generalization in deep network classifiers trained with the square loss." *Center for Brains, Minds and Machines* (*CBMM*) *Memo No. 112*, 2021.

[39] X. Y. Han, V. Papyan, and D. L. Donoho, "Neural collapse under mse loss: Proximity to and dynamics on the central path." *arXiv preprint arXiv:2106.02073*, 2021.

[40] J. Zhou, X. Li, T. Ding et al., "On the optimization landscape of neural collapse under mse loss: Global optimality with unconstrained features." *arXiv preprint arXiv:2203.01238*, 2022.

[41] S. Arora, Z. Li, and K. Lyu, "Theoretical analysis of auto rate-tuning by batch normalization." *CoRR*, abs/1812.03981, 2018.

[42] F. Anselmi, L. Rosasco, and T. Poggio, "On invariance and selectivity in representation learning." *CoRR*, abs/1503.05938, 2015.

[43] A. Ledent, Y. Lei, and M. Kloft, "Improved generalisation bounds for deep learning through l^∞ covering numbers." *CoRR*, abs/1905.12430, 2019.

[44] A Krizhevsky, "Learning multiple layers of features from tiny images." *Technical Report*, 2009.

[45] T. Salimans and D. P. Kingm, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks." *Advances in Neural Information Processing Systems*, 2016.

[46] T. Poggio and Y. Cooper, "Loss landscape: SGD can have a better view than GD." *Center for Brains, Minds and Machines (CBMM) Memo No. 107*, 2020.

[47] Q. Nguyen, "On connected sublevel sets in deep learning." In *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, pp. 4790–4799. PMLR, 2019.

[48] T. Liang, T. Poggio, A. Rakhlin, and J. Stokes, "Fisher-rao metric, geometry, and complexity of neural networks." *CoRR*, abs/1711.01530, 2017.

[49] S. Chatterjee, "Convergence of gradient descent for deep neural networks." *arXiv preprint arXiv:2203.16462*, 2022.

[50] M. Mohri, A. Rostamizadeh, and A. Talwalkar, "Foundations of Machine Learning". *MIT Press, Cambridge, MA, 2nd Edition*, 2018.

[51] N. Golowich, A. Rakhlin, and O. Shamir, "Size-independent sample complexity of neural networks." *CoRR*, abs/1712.06541, 2017.

[52] P. Rebeschini, "Algorithmic foundations of learning lecture 3: Rademacher complexity," *URL: https://www.stats.ox.ac.uk/ rebeschi/teaching/AFoL/20/material/slides03.pdf*, 2020.

[53] M. Xu, T. Poggio, and T. Galanti, "Complexity bounds for sparse networks." *Center for Brains, Minds and Machines (CBMM) Memo No. 1XX*, 2022.

[54] T. Galanti and T. Poggio, SGD noise and implicit low-rank bias in deep neural networks. *Center for Brains, Minds and Machines (CBMM) Memo No. 134*, 2022.

[55] T. Galanti, Z. S. Siegel, A. Gupte, and T. Poggio, "SGD and weight decay provably induce a low-rank bias in neural networks", *arXiv preprint arXiv:2206.05794*, 2022.

[56] O. Bousquet, S. Boucheron, and G. Lugosi, "Introduction to statistical learning theory." In *Summer School on machine learning*, pp. 169–207. Springer, 2003.

[57] Y. Jiang, D. Krishnan, H. Mobahi, and S. Bengio, "Predicting the generalization gap in deep networks with margin distributions." *arXiv preprint arXiv:1810.00113*, 2018.

[58] T. Evgeniou, M. Pontil, and T. Poggio, "Regularization networks and support vector machines." *Advances in Computational Mathematics*, vol. 13, pp. 1–50, 2000.

List of Figures

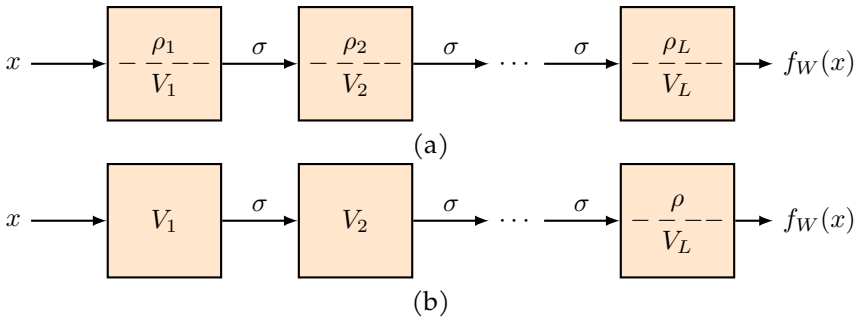


Figure 1: An illustration of two parametrizations of $f_W(x)$. In (a) we decompose each layer's weight matrix W_i into its norm ρ_i and its normalized version V_i. In (b) we normalize each layer except for the top layer's matrix W_L that is decomposed into a global ρ and the last layer V_L. Normalizing the weight matrices, as weight normalization (equivalent to LN) does, is different from Batch Normalization, though both normalization techniques capture the relevant property of normalization – to make the dot product invariant to scale.

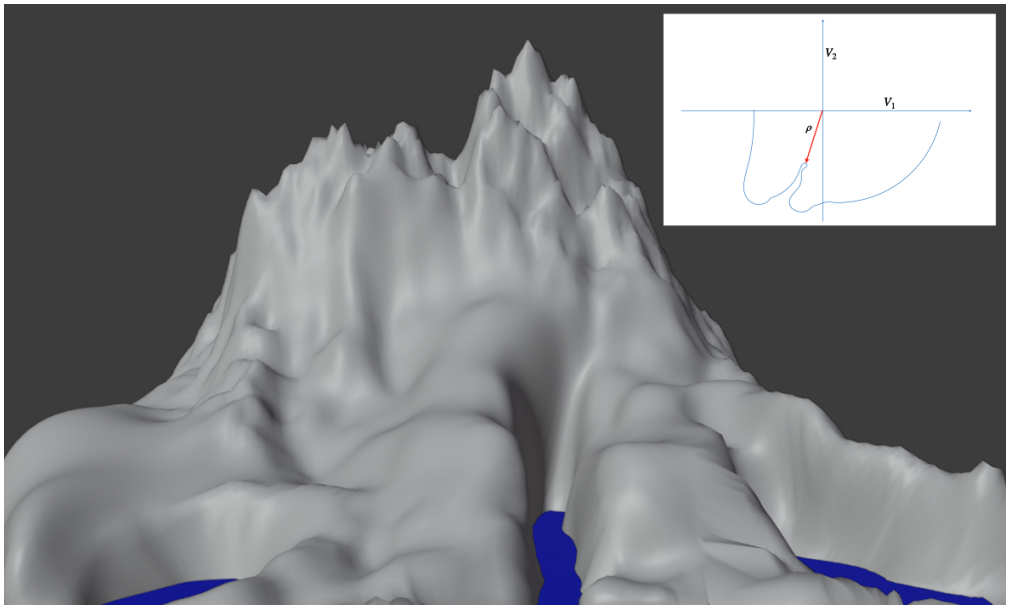
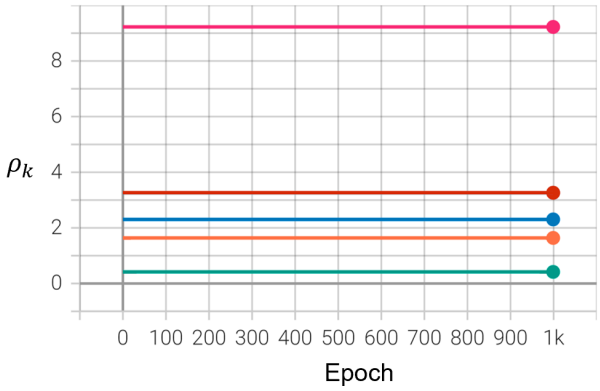


Figure 2: A speculative view of the landscape of the unregularized loss – that is for $\lambda = 0$. Think of the loss as the mountain emerging from the water with zero-loss being the water level. ρ is the radial distance from the center of the mountain as shown in the inset, whereas the V_k can be thought as multidimensional angles in this "polar" coordinate system. There are global degenerate valleys for $\rho \geq \rho_0$ with V_1 and V_2 weights of unit norm. The coastline of the loss marks the boundary of the zero loss degenerate minimum where $\mathcal{L} = 0$ in the high-dimensional space of ρ and $V_k \quad \forall k = 1, \cdots, L$. The degenerate global minimum is shown here as a connected valley outside the coastline. The red arrow marks the minimum loss with minimum ρ. Notice that, depending on the shape of the multidimensional valley, regularization with a term $\lambda\rho^2$ added to the loss, biases the solution towards small ρ but does not guarantee convergence to the minimum ρ solution, unlike the case of a linear network.



Color	Layer Name	ρ_k	Epoch No.
	Conv 1	1.639	999
	Conv 2	2.304	999
	Conv 3	3.262	999
	Conv 4	3.261	999
	FC 1	9.225	999
	FC 2	0.4179	999

Figure 3: Training dynamics of ρ_k during model (b) training with the Lagrange multiplier normalization over 1000 epochs. The model contains four convolutional layers, two fully connected layers and the top ρ (a learnable scalar parameter that can be initialized with different values). $\rho_k (k \in [L-1])$ are effectively stable during training because of weight normalization. The number of channels for the four convolutional layers (Conv1$\sim$Conv4) are 32, 64, 128 and 128, the filter size is 3×3, the hidden sizes of the last two fully connected layers (FC1 and FC2) are 1024 and 2, respectively. As mentioned in the text the norms of the convolutional layers is just the norm of the filters.

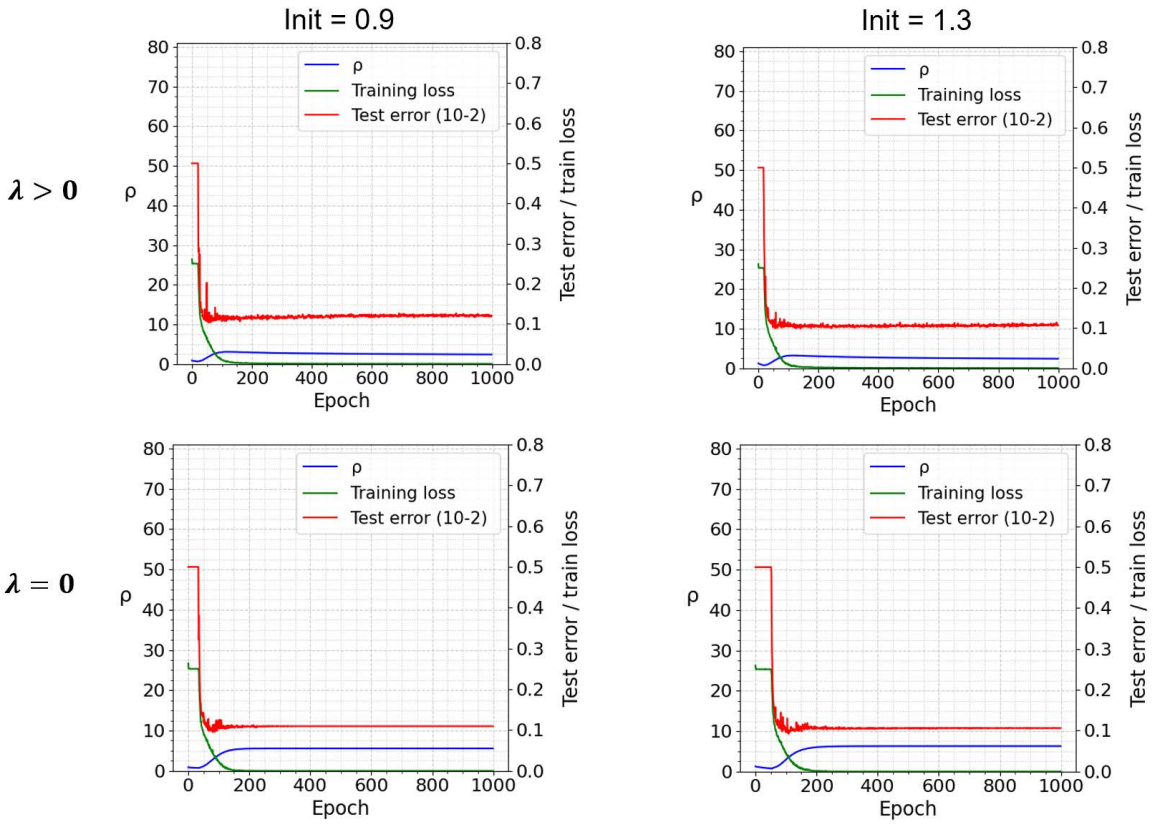


Figure 4: Training dynamics of ρ, of the training loss and of the test error over 1000 epochs with different initialization (0.9) in the first column and (1.3) in the second column.The number of channels for the four convolutional layers $(\text{Conv1}\sim\text{Conv4})$ are 32, 64, 128 and 128, the filter size is 3×3, the hidden sizes of the last two fully connected layers (FC1 and FC2) are 1024 and 2, respectively. The first row in the figure is with Weight Decay $\lambda = 0.001$, and the second row is with Weight Decay $\lambda = 0$. The network was trained with Cosine Annealing learning rate scheduler $(\text{with initial learning rate } \eta = 0.03,$ ending with $\eta = 0.0299)$.

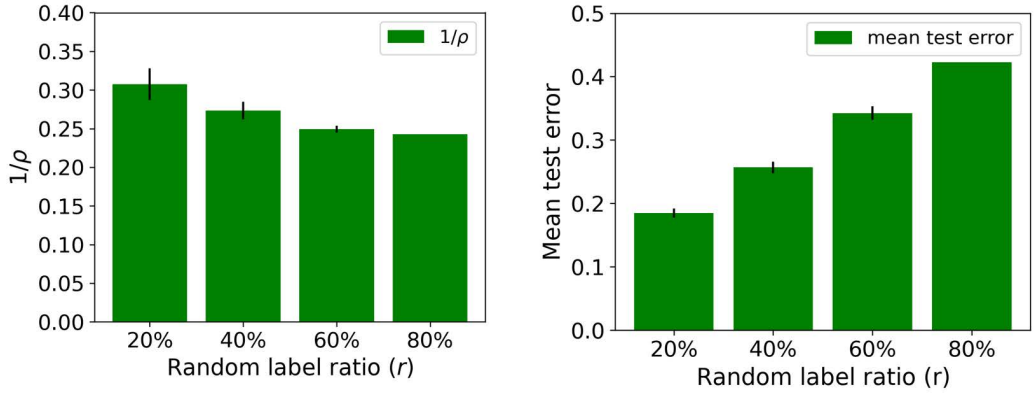


Figure 5: Mean $1/\rho$ and test error results over 10 runs for binary classification on CIFAR10 trained with Lagrange multiplier and different percentages of random labels ($r = 20\%, 40\%, 60\%$ and 80%), initialization scale 1 and weight decay 0.001. As mentioned in the text the norm of the convolutional layers is just the norm of the filters.(Note that this network fails to get convergence with 100% random labels.)

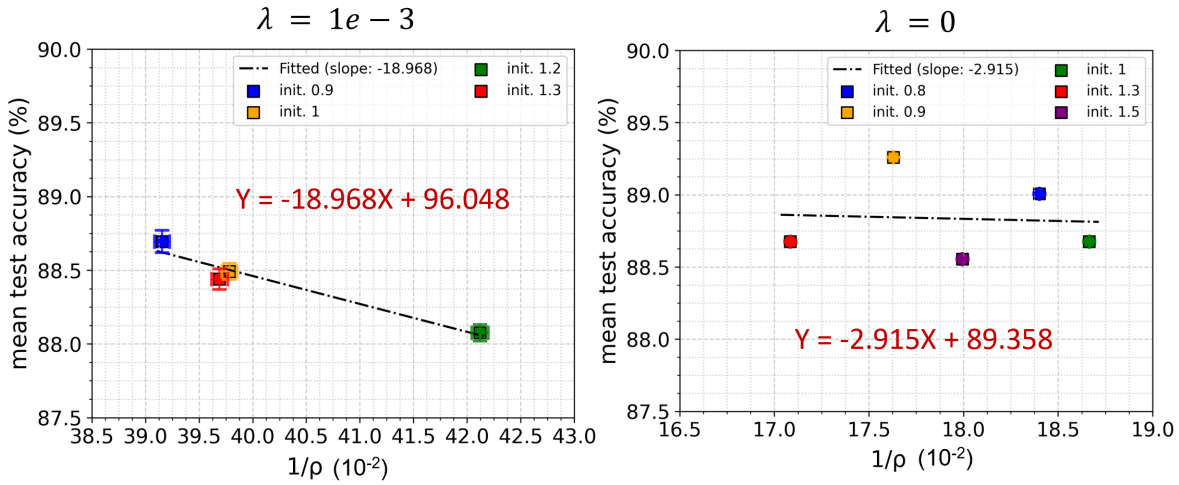


Figure 6: Scatter plots for $1/\rho$ and mean test accuracy based on 10 runs for binary classification on CIFAR10 using Lagrange multiplier normalization (LN), square loss and Weight Decay (left) and without Weight Decay (right). In the left figure, the network was trained with different initialization scales (init. $= [0.9, 1, 1.2, 1.3]$) and with weight decay ($\lambda = 1e-3$), while in the right figure, the network was trained with init. $= [0.8, 0.9, 1, 1.3, 1.5]$ and no weight decay ($\lambda = 0$). The horizontal and vertical error bars correspond to the standard deviations of $1/\rho$ and mean test accuracy computed over 10 runs for different initializations, while the square dots correspond to the mean values. When $\lambda > 0$, the coefficient (R^2), p-value and slope for linear regression between $1/\rho$ and mean test accuracy are: $R^2 = 0.94$, p-value = 0.031, slope = -18.968; When $\lambda = 0$, the coefficient $R^2 = 0.004$, p-value = 0.92 and the slope = -2.915.

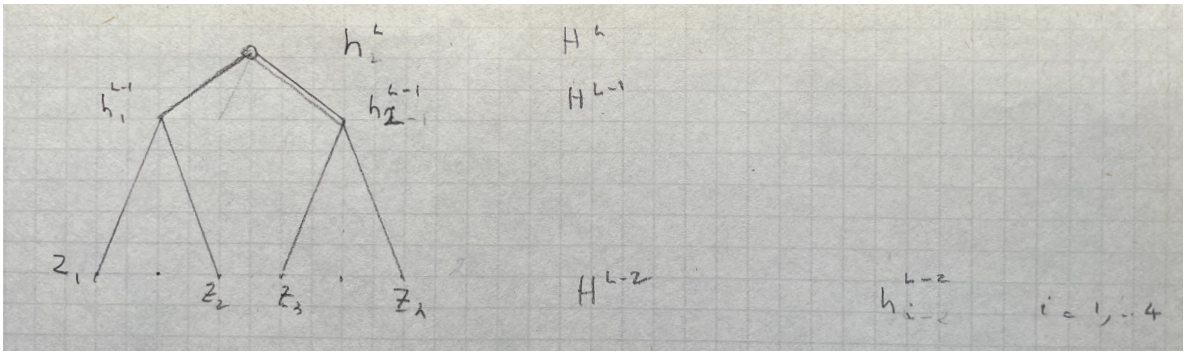
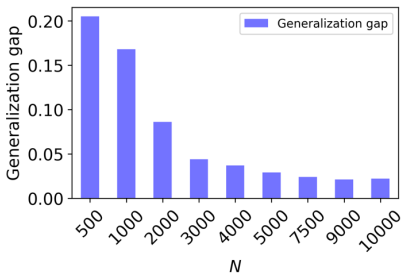


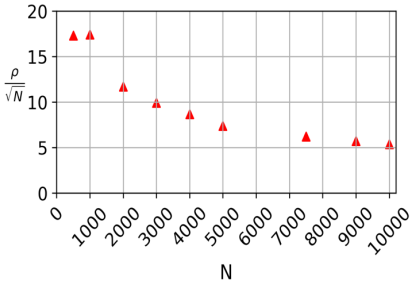
Figure 7: A binary CNN.

N	ρ	Train error	Test error	$\frac{\rho}{\sqrt{N}}$
500	387.672	0.000	0.205	17.3370
1000	551.282	0.000	0.168	17.4330
2000	523.698	0.000	0.086	11.7100
3000	544.018	0.006	0.050	9.9325
4000	550.868	0.008	0.045	8.7100
5000	522.870	0.011	0.040	7.3945
7500	540.055	0.012	0.036	6.2360
9000	544.738	0.011	0.032	5.7420
10000	538.174	0.014	0.036	5.3815

(a)



(b)

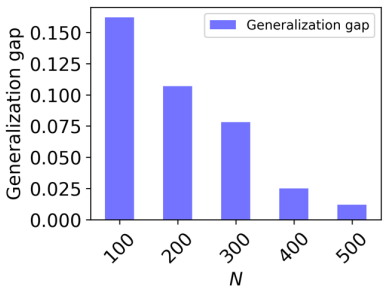


(c)

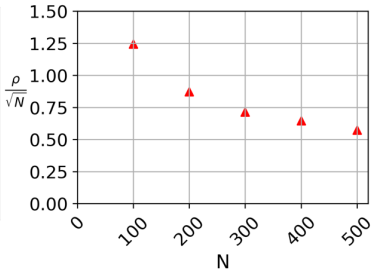
Figure 8: Product norm (ρ) and test error with respect to different training data sizes (N) for the six-layer model trained with LM and square loss. The initialization scale is 0.1, weight decay $\lambda = 10^{-3}$, no biases, the initial learning rate is 0.03 with cosine annealing scheduler; we used the SGD optimizer $(\text{momentum} = 0.9)$, test data size $= 2000$ in a binary classification task on CIFAR10 dataset. (a) The table shows the product norm ρ, mean training errors, mean test errors (average over the last 100 epochs), and generalization upper bound for different N. (b) A bar plot for the mean test errors by different N. (c) Generalization error upper bound defined as $(\frac{2\rho}{\sqrt{N}})$ for different N. The bounds are vacuous but "only" by an order of magnitude, while other bounds based on the number of parameters (here 3519335) are typically much looser.

N	total ρ	Train error	Test error	$\frac{\rho}{\sqrt{N}}$
100	12.425	0.127	0.289	1.243
200	12.351	0.122	0.229	0.873
300	12.359	0.136	0.214	0.714
400	12.894	0.163	0.188	0.645
500	12.802	0.165	0.177	0.573

(a)



(b)

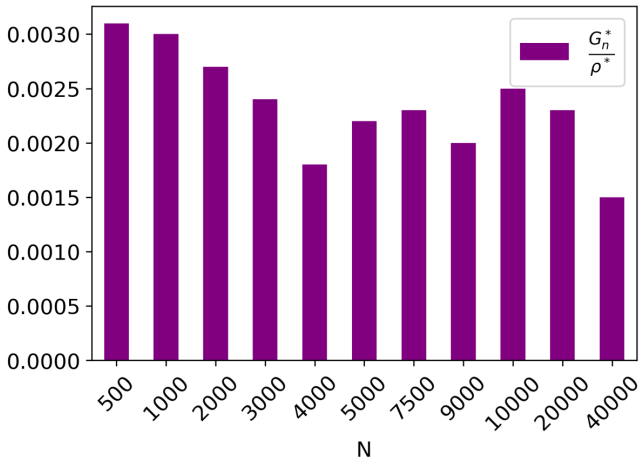


(c)

Figure 9: Product norm (ρ) and test error with respect to different training data sizes (N) for the three-layer model (with non-overlapped convolutional image patches, kernel size $= 3 \times 3$, stride $= 3$) trained with LM and square loss. The initialization scale is 0.1, weight decay $\lambda = 0.001$, no biases, batch size is 32, the initial learning rate is 0.03 with cosine annealing scheduler; we used the SGD optimizer (momentum $= 0.9$), test data size $= 2000$ in a binary classification task on CIFAR10 dataset. (a) The table shows the product norm ρ, mean training errors, mean test errors (average over the last 100 epochs), and generalization upper bound for different N . (b) A bar plot for the mean test errors by different N . (c) Generalization error upper bound is a constant (see text) times $(\frac{\rho}{\sqrt{N}})$. The bounds are almost not vacuous depending on the constant (see text).

N	ρ^*	ρ	G_n^*	G_n	$\frac{G_n^*}{\rho^*}$	$\rho\frac{G_n^*}{\rho^*}$
500	146.811	99.752	0.459	0.182	0.0031	0.3119
1000	134.972	94.775	0.400	0.116	0.0030	0.2809
2000	107.449	71.349	0.293	0.032	0.0027	0.1946
3000	94.316	70.617	0.231	0.022	0.0024	0.1730
4000	90.940	67.772	0.164	0.018	0.0018	0.1222
5000	77.311	68.531	0.169	0.020	0.0022	0.1498
7500	75.338	68.026	0.170	0.017	0.0023	0.1535
9000	70.696	68.874	0.144	0.013	0.0020	0.1403
10000	67.653	67.908	0.166	0.014	0.0025	0.1666
20000	35.116	63.577	0.081	0.001	0.0023	0.1466
40000	32.765	63.359	0.048	0.003	0.0015	0.0928

(a)



(b)

Figure 10: Generalization gap and product norm results based on 5-layer network trained with different number of training data samples (N) for binary classification of CIFAR10. (a) ρ^* and G_n^* represent the *empirical* product norms and generalization gaps w.r.t. N measured by the model trained with 100% random labels; ρ and G_n indicate the product norms and generalization gaps w.r.t. N measured by the model trained with natural labels. The entire model was optimized with LM and Weight Decay $(\lambda = 10 - 3)$, and was trained with initial learning rate 0.03, initialization scale 0.1 and 10000 epochs. (b) A bar plot of G_n^*/ρ^* for different number of training data samples N.

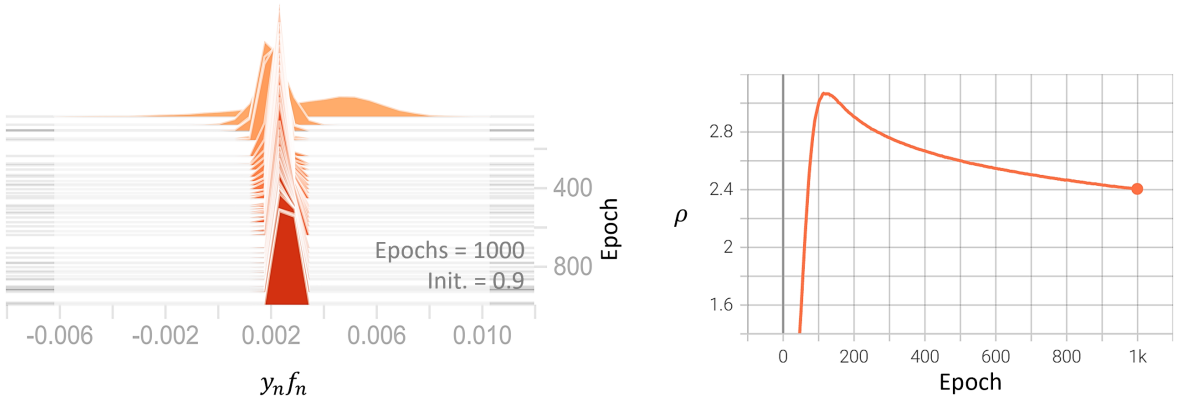


Figure 11: Histogram of $y_n f_n$ across 1000 training epochs for binary classification on the CIFAR10 dataset with Lagrange multiplier and weight decay $(\lambda) = 0.001$, initial learning rate 0.03, initialization 0.9. The histogram narrows as training progresses. The final histogram (in red) is concentrated, as expected for the emergence of NC1. The right side of the plot shows the time course of the top ρ over the same 1000 epochs.

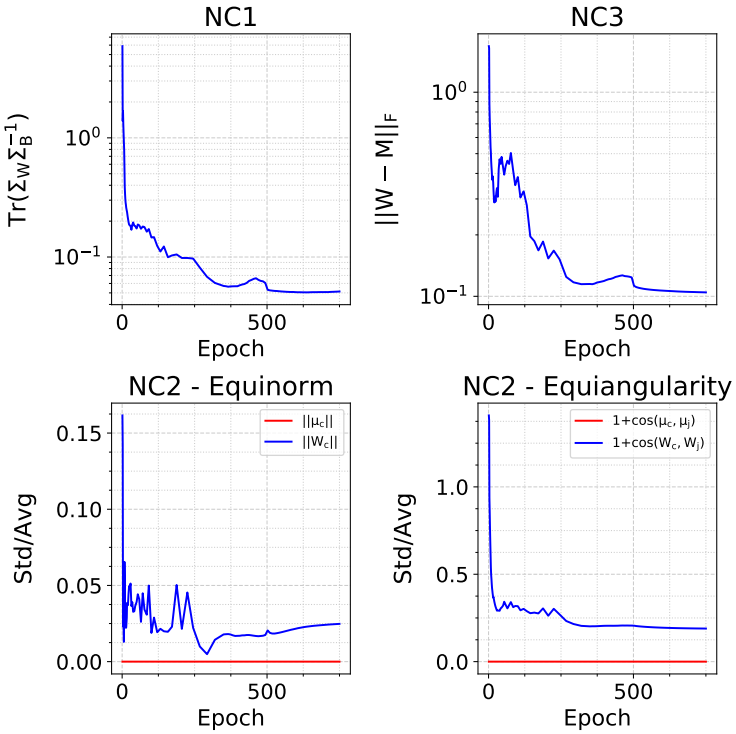


Figure 12: Neural Collapse occurs during training for binary classification. The key conditions for Neural Collapse are: (i) NC1 - Variability collapse, which is measured by $\mathrm{Tr}(\Sigma_W\Sigma_B^{-1})$, where Σ_W, Σ_B are the within and between class covariances, (ii) NC2 - equinorm and equiangularity of the mean features $\{\mu_c\}$ and classifiers $\{W_c\}$. We measure the equinorm condition by the standard deviation of the norms of the means (in red) and classifiers (in blue) across classes, divided by the average of the norms, and the equiangularity condition by the standard deviation of the inner products of the normalized means (in red) and the normalized classifiers (in blue), divided by the average inner product, and (iii) NC3 - Self-duality or the distance between the normalized classifiers and mean features. This network was trained on two classes of CIFAR10 with Weight Normalization and Weight Decay = 5e-4, learning rate 0.067, for 750 epochs with a stepped learning rate decay schedule.

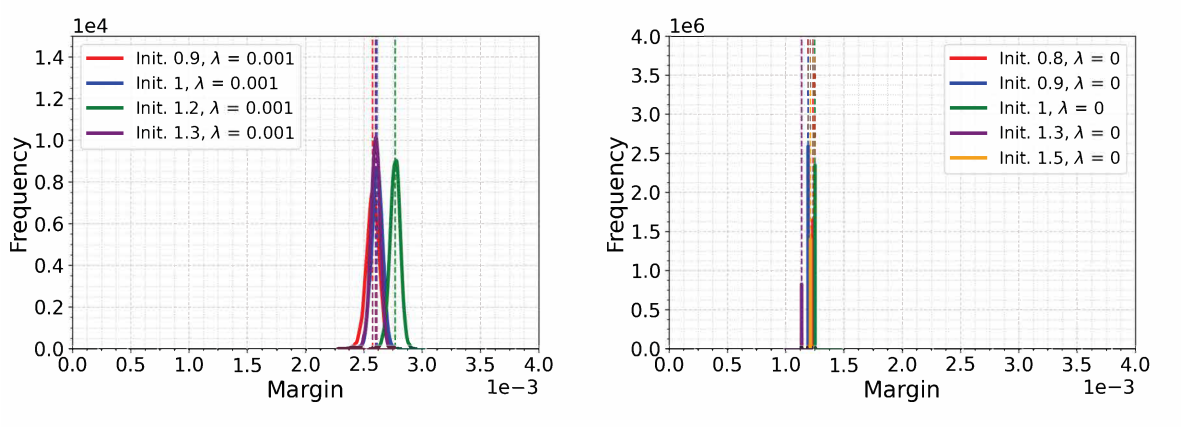


Figure 13: Training margins computed over 10 runs for binary classification on CIFAR10 trained with square loss, Lagrange multiplier normalization, and Weight Decay $(\lambda) = 0.001$ (left) and without Weight Decay (right, $\lambda = 0$) for different initializations ($init.$ = 0.8, 0.9, 1, 1.2, 1.3 and 1.5) with SGD and minibatch size of 128. The margin distribution is Gaussian-like with standard deviation $\approx 10^{-4}$ over the training set ($N = 10^4$). The margins without Weight Decay result in a range of smaller margin values, each with essentially zero variance. As mentioned in the text the norms of the convolutional layers is just the norm of the filters.