CENTER FOR
Brains
Minds+
Machines

CBMM Memo No. 117 **April 30, 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

Here we consider a model of the dynamics of gradient flow under the square loss in overparametrized ReLU networks. We show that convergence to a solution with the absolute minimum ρ, which is the product of the Frobenius norms of each layer weight matrix, is expected when normalization by a Lagrange multiplier (LN) is used together with Weight Decay (WD). We prove that SGD converges to solutions that have a bias towards 1) large margin (i.e. small ρ) and 2) low rank of the weight matrices. In addition, we predict the occurence of Neural Collapse without *ad hoc* assumptions such as the unconstrained features hypothesis.

This is effectively an update of CBMM Memo 112, with some of the same material, more figures on experiments and a better discussion of generalization.

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 non-vacuous 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

May 23, 2022

Abstract

Recent results of [1] suggest that square loss performs on par with cross-entropy loss in classification tasks with deep networks. While the theoretical understanding of training deep networks with the cross-entropy loss has been growing ([2] and [3]) in terms of margin maximization, the study of square loss for classification has been lagging behind. Here we consider a specific model of the dynamics of, first, gradient flow (GF), and then SGD, in overparametrized ReLU networks under the square loss. Under the assumption of convergence to zero loss minima, we show that solutions have a bias toward small ρ, defined as the product of the Frobenius norms of each layer 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). For $\lambda \to 0$ the solution would be the interpolating solution with minimum ρ. 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 GD dynamics introduced by carefully chosen close-to-zero initial conditions on the norms of the weights, similar to the case of overparametrized linear networks (see Appendix E). However, for $\lambda = 0$ we often observe solutions with large ρ that are suboptimal and probably in the NTK regime.

We show that convergence to an ideal equilibrium of SGD ($\dot{V}_k = 0$ for all minibatches) with $\lambda > 0$ would imply rank one weight matrices. This is impossible generically, implying that SGD never converges to the same set of V_k across all minibatches. We claim that this is the origin of an asymptotic SGD noise. The SGD noise disappears when $\lambda = 0$, when the loss function is the square loss. It does not disappear, even in the absence of regularization, when an exponential-type loss is used if weight normalization is used. The same rank one constraint implies that SGD has a bias towards low rank solutions depending on λ and on minibatch size.

We also prove that quasi-interpolating solutions obtained by gradient descent in the presence of WD show the recently discovered behavior of Neural Collapse (NC) [4]. Normalization by Lagrange multiplier is identical to weight normalization but different from the more commonly used and better performing batch normalization.

For $\lambda > 0$ a main property of the minimizers that bounds their expected error is ρ which is the inverse of the margin: we prove that among all the quasi-interpolating solutions (for $\lambda > 0$), the ones associated with smaller ρ have better bounds on the expected classification error. Our bounds on the expected error for quasi-interpolating solutions based on ρ turn out to be non-vacuous. This is encouraging because in our experiments there is overparametrization and thus quasi-interpolation of the training data but sufficiently small Rademacher complexity wrt the number of training data.

In summary, this paper describes how gradient descent on the square loss in the presence of regularization can converge to minimum ρ solutions, corresponding to max margin solutions, and show that they are biased towards small rank of the weight matrices. Associated with these solutions are properties such as Neural Collapse and empirically non-vacuous upper bounds on the generalization error. Our analysis of the square loss suggests a bias of SGD in the presence of weight decay towards minimum ρ and low rank solutions.

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 [2, 3], 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 [5, 2, 6]. Recently however, [1] 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) [7]. Can a theoretical analysis explain when and why regression should work well for classification? This question was the original motivation for previous versions of this paper [8]. The present paper provides a substantial update.

In deep learning, 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). Although all the interpolating solutions are optimal solutions of the regression problem, they will in general correspond to different 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 of the regression problem obtained by GD to have 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 weight decay. We describe the qualitative dynamics of the the deep network parameters and show that ρ which is the product of the Frobenius norms of the weight matrices, grows non-monotonically until small ρ, that is large margin, solutions are reached. Assuming that local minima and saddle points can be avoided, this analysis shows that with small initialization and weight decay there will be convergence to a global minimum with a ρ biased to be small. In the following, we will occasionally refer to ρ as the "norm" of the function represented by the deep ReLU network.

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, depending on the regularization (also called weight decay) parameter, there is a stronger independence from initial conditions, as has been observed in [1]. We show that weight decay helps stabilize normalization of the weights, in addition to its role in the dynamics of the norm.

We then describe how to extend our model to include gradient descent. A comparison of normalization using Lagrange Multipliers (LM) or Weight Normalization (WN) or Batch Normalization (BN) is particularly interesting for explaining the role of normalization in training deep networks and the differences between normalization techniques.

Next, 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 a SGD bias towards small rank solutions that reinforces a similar bias due to maximization of the margin under normalization (that can be inferred from [11]).

Next we show that these quasi-interpolating solutions satisfy the recently discovered Neural Collapse (NC) phenomenon [4]. 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[1].

Finally, we apply basic bounds on expected error to the solutions provided by SGD (for $\lambda > 0$), which have minimum ρ. The bounds turn out to be non-vacuous. This is encouraging because in our experiments there is overparametrization and thus quasi-interpolation of the training data but sufficiently small Rademacher complexity wrt the number of training data.

[1] We discuss its extension to exponential-type loss functions in an Appendix.

Contributions The main contributions in this paper are

- In Section 3, we analyze the dynamics of deep network parameters, their norm, and the margins under gradient flow on the square loss, using *Lagrange normalization* (LM). We describe the evolution of the norm, and the role of Weight Decay and normalization in the training dynamics. The analysis is extended to the case of Gradient Descent in Section 5.
- For SGD, we show that when $\lambda > 0$ there is always SGD noise, even asymptotically, and associated with it, there is a bias towards solutions of low rank – depending on the size of the minibatches.
- We show that under certain assumptions, critical points of Stochastic Gradient Descent with Weight Decay satisfy the conditions of Neural Collapse for the square loss functions. Our proof technique also allows us to find the relationship between the Simplex ETF and the margin of the solution.
- Our conclusions and theoretical observations are supported by experiments.

Outline We structure the rest of the paper as follows. We start describing related work. In section 3 we formulate a model that assumes three simplifying assumptions. For this model we analyze the dynamics of gradient flow under the square loss for a binary classification problem. We use an analysis of its continuous dynamics to illuminate the role of Weight Decay and Batch/Weight Normalization in deep learning. We extend in section 5 our model from Gradient Flow to an approximation of Gradient Descent, eliminating our second simplifying assumption. We then turn to analyze SGD, the associated noise and its bias towards low-rank solutions. In section 7 we use our analysis of the dynamics in the binary classification case to justify an assumption of margins being very close to each other at convergence to predict the phenomenon of Neural Collapse when training on the square loss. A simple generalization bound links margin, which is the inverse of ρ, to expected error, in particular in the interpolation or quasi-interpolation case when the empirical error is zero or close to zero. In our experiments the bound is not vacuous even in our overparametrized case. Throughout we describe experiments on CIFAR10 that illustrate several of the theoretical results. We conclude in section 9 with a discussion of our results and their implications for generalization.

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 [12] and for deep networks in [2, 3, 13, 14]. Recent interest in using the square loss for classification has been spurred by the experiments in [1], though the practice of using the square loss is much older [7]. Muthukumar et. al. [15] 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 [8]). Recent work also studied interpolating kernel machines [16, 17] 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 [18, 19] 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 [20, 21, 22]. Detailed analyses of two-layer neural networks such as [23, 24, 25] typically use the square loss as an objective function. However these papers do not specifically consider the task of classification.

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 [11]). 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 [11]) showed that replacing the weight matrices 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.

Neural Collapse (NC) [4] 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. Mixon et. al. [26] show NC in the regime of "unconstrained features". Other papers have shown the emergence of NC when using the cross-entropy loss [27, 28, 29]. While preparing this paper, we became aware of recent results by Ergen and Pilanci [30] (see also [31]) who derived NC for the square loss, through a convex dual formulation of deep networks. Our independent derivation is different. Unlike most proofs we do not make any assumption and derive NC[2] from an analysis of convergence for the square loss.

3 Problem Setup

In this section, we describe the training settings considered in our work. We study training deep convolutional neural network with ReLU non-linearity using square loss minimization for classification problems. In the proposed analysis, we apply two types of normalization techniques (Lagrange Multiplier and Batch Normalization) during training as well as regularization (also called Weight Decay), since such mechanisms seem essential for reliably training deep networks using gradient descent [32, 1].

3.1 Assumptions

Throughout the theoretical analysis we make in some places simplifying assumptions relative to standard practice in deep neural networks. First, **(i)** we mostly consider the case of binary classification. In addition, **(ii)** we consider in the first part continuous Gradient Flow (GF) instead of Stochastic Gradient Descent (SGD). Gradient flow is the limit of discrete Gradient Descent algorithm with the learning rate being is infinitesimally small. Furthermore, **(iii)** we assume normalization to be carried out 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 from the beginning that the network is overparametrized and that that there is convergence to global minima with appropriate initialization, parameter values and data. A recent analysis [33] provide a powerful new criterion for convergence, assuming that the input dimension is greater than the number of data points.

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 $f_W(x) = W_L \sigma\left(W_{L-1} \ldots \sigma\left(W_1 x\right) \ldots\right)$, 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 $f_W(x) = \rho_L V_L \sigma\left(\rho_{L-1} \ldots \sigma\left(\rho_1 V_1 x\right) \ldots\right)$. 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 $f_W(x) = \rho f_V(x) = \rho V_L \sigma\left(V_{L-1} \ldots \sigma\left(V_1 x\right) \ldots\right)$. 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 $f_n := f_V(x_n)$ (the network of Figure 1b, evaluated on the training sample x_n).

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

[2] There is a gap, however, in the proof, see later.

[3] We choose the Frobenius norm here to simplify our calculations. While a different choice of norm (such as ℓ_1) may help prove tighter generalization bounds, they are functionally equivalent for the purpose of analyzing dynamics and the margin (as noted in Section 3.4 of [34]).

unit variance. WN normalizes the weight matrices (section 10 in [6]). 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. In Appendix F, we discuss differences between the normalization algorithms (LM and BN).

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 during gradient descent 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 1b).

Minimization of the regularized loss function $\frac{1}{N}\sum_n(\rho f_n - y_n)^2 + \lambda\rho^2$ under the constraint $\|V_k\|^2 = 1$ can be summarized in the following manner

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

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

Notations. Throughout the paper, we use the following set of notations. For an integer $k \geq 1$, $[k] = \{1, \ldots, k\}$. For any real vector z, $\|z\|$ denotes its Euclidean norm. For a finite set A, we denote by $U[A]$ the uniform distribution over A. Let Q be a distribution over $X \subset \mathbb{R}^d$ and $u : X \to \mathbb{R}^p$. We denote by $\mathrm{Var}_u(Q) = \mathbb{E}_{x\sim Q}[\|u(x) - \mu_u(Q)\|^2]$ variance of $u(x)$ for $x \sim Q$.

Separability and Margins. Two important aspects of classification is 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, $\forall n: \ \bar{f}_n > 0$. Furthermore, when $\sum_n \bar{f}_n > 0$, we say that *average separability* is satisfied. Notice that the minimum of $\mathcal{L}(\rho, \{V_k\}_{k=1}^L)$ for $\lambda = 0$ is zero and 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 $M \geq 0$ and $\sigma^2 = \sum_n(\bar{f}_n - \mu)^2 \geq 0$ are not negative.

Interpolation and Quasi-interpolation. Assume that the weights V_k are normalized at convergence. Then $\mathcal{L} = \frac{1}{N}\sum_n(1 - \rho\bar{f}_n)^2 + \lambda\rho^2 = 1 - 2\rho\mu + \rho^2 M + \lambda\rho^2$. For $\lambda = 0$, $\mathcal{L} = 0$ implies $\mu = \bar{f}_n, \quad \forall n$ and $\mu = \frac{1}{\rho}$. However, for $\lambda > 0$ the assumption that all $\bar{f}_n$ are equal yields $M = \mu^2$ and thus $\mathcal{L} = \rho^2\mu^2 - 2\rho\mu + (1 + \lambda\rho^2)$. Setting $\mathcal{L} = 0$ gives a second order equation which does not have real solution for $\rho \geq 0$ and $\mu \leq 1$. Thus for $\lambda > 0$ there are no solutions that have the same margin for all points and reach zero loss. However, solutions that have the same margin for all points and correspond to zero gradient wrt ρ exist. Assuming $\sigma = 0$, setting the gradient of $\mathcal{L}$ wrt ρ equal to zero yields $\rho\mu^2 - \mu + \lambda\rho = 0$ which gives

$$\mu = \frac{1 \pm \sqrt{1 - 4\lambda\rho^2}}{2\rho}. \tag{2}$$

If $4\lambda\rho^2 < 1$, then solutions exist such that $\rho\mu < 1$. This solution is not interpolating.

[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.

Experiments Our binary classification experiments were performed using the standard CIFAR10 dataset [35]. 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 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). Following each convolution layer, we applied a ReLU nonlinear activation function and Weight Normalization (WN) to all layers, freezing the weights of the WN parameter "g" and normalizing the "v" matrix at each layer w.r.t. its Frobenius norm, such that the norm (ρ_k, $k \in [L]$) of the weight matrix remains constant across all layers except for the top ρ.

We also performed a separate set of experiments with Batch Normalization with the same architecture and also with a different architecture, without any densely connected layer. These experiments are described in Appendix F.2.1. They are similar to the results with weight normalization (or Lagrange Multipliers).

4 Training Dynamics

4.1 Gradient Flow Equations

The gradient flow equations are as follows

$$\dot{\rho} = -\frac{\partial \mathcal{L}}{\partial \rho} = \frac{2}{N}\sum_n (1-\rho \bar{f}_n)\bar{f}_n - 2\lambda\rho \;\text{ and }\; \dot{V}_k = -\frac{\partial \mathcal{L}}{\partial V_k} = \frac{2}{N}\sum_n (1-\rho\bar{f}_n)\rho\frac{\partial \bar{f}_n}{\partial V_k} - 2\nu_k V_k. \tag{3}$$

In the second equation we can use the unit norm constraint on the $\|V_k\|$ to determine the Lagrange multipliers ν_k. Using Lemma 6 (Appendix A), the constraint $\|V_k\|^2 = 1$ implies $\frac{\partial \|V_k\|^2}{\partial t} = V_k^T \dot{V}_k = 0$, 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

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

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 in addition $\sum_n \ell_n \frac{\partial f_n}{\partial V_k} - V_k f_n = 0$, then $\dot{V}_k = 0$.

As a consequence of the above derivation, we can represent the loss function in terms of ρ, $\dot{\rho}$ and μ in the following manner.

Lemma 1 *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 in the following manner $\mathcal{L}(\rho, \{V_k\}_{k=1}^L) = 1 - \rho(\frac{1}{2}\dot{\rho} + \mu)$.*

Proof First, we consider that

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

where the second equations follows from $\|V_k\| = 1, \quad k = 1, \cdots, L$ 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 $\mathcal{L} = 1 - \frac{1}{2}\rho\dot{\rho} - \rho\mu = 1 - \rho(\frac{1}{2}\dot{\rho} + \mu)$ as desired. ■

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 4.2, 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 tends to zero.

Lemma 2 *Let $f_W(x) = \rho f_V(x)$ be a neural network. Assume that during training time $\lim_{t\to\infty} \mathcal{L}(\rho, \{V_k\}_{k=1}^L) = 0$ 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 1, $\rho\dot{\rho} \to 0$. 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 $\lambda = 0$) $\dot{V}_k \to 0$. ■

4.2 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 are connected in a single zero-loss degenerate valley for $\rho \geq \rho_0$. Figures 3 and 4 show 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 is connected (under certain assumptions), the absolute minimum ρ point 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 Figure 4).

The degeneracy of the global minimizers achieving zero loss of

$$L(f_W) = \sum_{i=1}^n (f_W(x_i) - y_i)^2 \tag{8}$$

was established recently under the assumption that the network f is overparametrized with a number of weights $d \gg n$ and that is able to interpolate the training data achieving $\min_W L(f_W) = 0$ which implies $\forall i \in [n] : \ f_W(x_i) = y_i$.

If we assume overparametrized networks with $d \gg n$, where d is the number of parameters and n is the number of data points Cooper (see references in [36]) proved that the global minima of $L(w)$ are highly degenerate[5] with dimension $d - n$.

Theorem 1 ([36]) *We assume an overparametrized deep 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 [37]:

Theorem 2 ([37]) *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*. The reason is that the connected valley will in general twist in the space of parameters in such a way that following it does not monotonically increase or decrease ρ.

We expect for large ρ (with $\lambda = 0$), 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 for $\lambda > 0$ there is a "pull" towards the minimum ρ_0 within the local degenerate minimum of the loss.

4.3 Qualitative 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, the fact that matrix norms are sub-multiplicative, and $\|V_k\| = \frac{\partial \mu}{\partial V_L}$. 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 is typically 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 $f(x) = 0$ (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 very small ρ, that is $\rho \ll \rho_0$. 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 $f_n > 0 \quad \forall n$, 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.

Following Lemma 1, 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

[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 under-determined case $n < d$, there are no zero-dimensional components of V , and thus V is either empty or infinite".

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 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 (see Appendix 4.2). 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. Notice that all ρ_k of Figure 1 are constant because of weight normalization, see Figure 5.

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 $\rho|f_n| \approx 1$. 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 remind of the Neural Tangent Kernel (NTK) solutions [22], where the parameters do not move too far from their initialization. A formal version of the same argument is based on the following result.

Lemma 3 *Let $\{V_k\}_{k=1}^L$ be a set of weights in the last layer of the network of Figure 1, such that, $\forall n \in [N]: \ \bar{f}_n = \mu^*$. Then, for any unit-norm vector $V_L' \in \mathbb{R}^p$, such that, $V_L^T V_L' = \alpha \in (0,1)$ the neural network $\hat{f}(x) = (V_L')^\top h(x)$ has a margin $\alpha\mu^*$.*

Proof Let $f_n = f(x_n)$, $h_n = h(x_n)$, V_L' be a unit-vector, such that, $V_L^T V' = \alpha$ and $\hat{f}_n = \hat{f}(x_n) = (V_L')^\top h_n$. We recall that $y_n V_L^\top h_n = y_n f_n = \bar{f}_n = \mu^*$. In particular, by multiplying both sides by $(V_L')^\top V_L = \alpha$, we obtain the following equation $y_n \hat{f}_n = y_n (V_L')^\top h_n = y_n (V_L')^\top V_L V_L^\top h_n = \alpha y_n f_n = \alpha \bar{f}_n = \alpha\mu^*$, which concludes the proof. ■

Thus if a network exists providing an interpolating solution with a minimum ρ, there exist networks that differ only in the last V_L layer which also interpolate with larger ρ. As a consequence there is a continuum of other solutions that differ only in the weights V_L of the last layer:

Corollary 1 *Assume that the interpolating solution with minimum ρ satisfies (with a small $\lambda > 0$) NC1 (see later) and NC3. Then there is a continuous of other solutions with larger ρ that satisfy NC1 but not NC3.*

Notice that there may be interpolating solutions corresponding to different sets of weights in layers below L to which the above statement does not apply. The corollary can be read as stating that there is a valley of minimizers starting from a zero-loss minimizer which has neural collapse for a certain ρ. Similar arguments could be made for other weight matrices using the fact [38] that the set of all m, n rectangular real matrices of rank r is connected by analytic regular arcs when $m > n = r$.

In Figure 6 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 6 respectively. The top row plots also show that Weight Decay makes the final solutions robust to the scale of initialization, in terms of ρ and of the train loss. This robustness is seen more clearly in Figure 7, where we plot the training margins $(y_n f_n)$ obtained with and without Weight Decay. In the right plot, without Weight Decay, the margin distributions depend on the initialization, while in the left plot they cluster around roughly the same value.

Summary 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 6, upper and lower row). This dynamics takes place *even in the absence of regularization that is for $\lambda = 0$*; however, as we will see, $\lambda > 0$ makes the process more robust and bias it towards small ρ.

To explain more in detail the initial dynamics of the gradient flow let us neglect here the dynamics of normalization; thus we assume $\|V_k\| = 1$ at all times. This is tantamount to neglecting fluctuations due to normalization in the dynamics of V_k or to assuming that the normalization dynamics is fast (wrt to ρ dynamics)). Then

Observations

- *Immediately after initialization with $\rho(0) < \rho_0$, if there is convergence (depending on parameters values and network architecture, convergence is not guaranteed) ρ grows non-monotonically until $\rho(t)$ is within a neighborhood of ρ_{eq} (within $\lambda\rho_0 + \sum f_n^2$).*
- *There may be oscillations in $\rho(t)$, that is time intervals in which $\dot{\rho} < 0$: these intervals are limited.*
- *$\dot{\rho} = 0$ between positive and negative values may be a critical point of the system if the $\dot{V}_k = 0$ for the associated ρ give compatible f values; these critical points may be local minima or saddle points; if local minima and saddles can be avoided (by SGD or by restarting optimization since $\mathcal{L} > 0$) the dynamics will eventually converge to an interpolation or quasi-interpolation solution with $\mathcal{L} = 0$ if $\lambda = 0$ and otherwise close to zero.*
- *Weight decay performs the traditional role of promoting solutions with small norm. In the case of large initialization, we can see from (7) that ρ_{eq} is determined by λ.*
- *Norm regularization is however not the only contribution of weight decay. The critical points $\dot{V}_k = 0$ may not be normalized properly – and thus may not impose a rank one constraint, see later – if the solution interpolates. In particular, an unnormalized interpolating solution can satisfy the equilibrium equations for $\dot{V}_k$. This is expected from the constrained dynamics which by itself constrains the norm of the V_k to not change during the iterations[6]. By preventing exact interpolation, weight decay ensures that the critical points of the V_k dynamics lie on a fixed Frobenius norm ball. As we will discuss later, by preventing exact interpolation, gradient flow with weight decay under the square loss shows the phenomenon of SGD noise and of Neural Collapse.*

Additional remarks are that $\mathcal{L}$ 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 (4.1) suggests that for small ρ, the term dominating the decrease in $\mathcal{L}$ is $-2\rho\mu$. For larger ρ, the term $\rho^2 M = \rho^2(\sigma^2 + \mu^2)$ becomes important: eventually $\mathcal{L}$ 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 $\mathcal{L} = 0$ (since $\mu = \frac{1}{\rho_0}$, $M = \mu^2$, $\sigma^2 = 0$).

In all these observations we have assumed gradient flow. The dynamics for gradient descent implies by itself the presence of damped oscillations[7] in ρ. In addition, the randomness of SGD also contributes to transient decreases in the norm ρ.

5 Extending the Analysis to GD

Recently, [39] showed that a natural approximation to gradient descent within a continuous gradient flow formulation is equivalent to adding to the loss functional $\mathcal{L}$ a term proportional to $\frac{\eta}{4}$, consisting of the norm square of the gradient of $\mathcal{L}$. This is equivalent (see [40]) to replacing in the gradient flow equation terms like $\dot{x}$ with terms that are $\frac{\eta}{2}\ddot{x} + \dot{x}$. The informal explanation is that the gradient descent term $x(t+\eta) - x(t) = -\eta F$ can be approximated by expanding $x(t+\eta)$ in a Taylor series for small η to a quadratic approximation, that is $x(t+\eta) \approx x(t) + \eta\dot{x}(t) + \frac{\eta^2}{2}\ddot{x}(t)$. Thus the gradient descent equation becomes $\dot{x}(t) + \frac{\eta}{2}\ddot{x}(t) = -F$.

With this approximation Equations (3) become (taking into account that $\mathcal{L} = \sum_n (1 - \rho\bar{f}_n)^2 + \nu \sum_{k=1}^{L-1} \|V_k\|^2 + \lambda\rho^2$)

[6] Numerical simulations show that even for linear degenerate networks convergence is independent of initial conditions only if $\lambda > 0$. In particular, normalization is then effective at ρ_0, unlike the $\lambda = 0$ case.

[7] As we will show later, discretization is similar to adding a second derivative term $\eta\ddot{\rho}$ to the dynamical system which introduces by itself oscillations in the dynamics of ρ.

$$\frac{\eta}{2}\ddot{\rho} + \dot{\rho} = 2[\sum_n (1-\rho\bar{f}_n)\bar{f}_n] - 2\lambda\rho \tag{9}$$

$$\frac{\eta}{2}\ddot{V_k} + \dot{V_k} = 2\rho\sum_n [(1-\rho\bar{f}_n)(V_k\bar{f}_n - \frac{\partial \bar{f}_n}{\partial V_k})] \quad \forall k \leq L. \tag{10}$$

We observe immediately that the equation in ρ – since it has the form $\eta\ddot{\rho} + \dot{\rho} + 4\lambda\rho - 4C = 0$ may show oscillations(the frequency of the "undamped oscillations" is $\sqrt{\frac{4(\frac{1}{N}\sum \bar{f}_n^2 + 2\lambda)}{\eta}}$. Whenever $1 > \sqrt{\frac{2}{N}\sum_n \bar{f}_n^2 + 2\lambda}$, the linearized dynamics is the dynamics of a damped oscillator. However, we didn't find empirical evidence for such oscillations examining the power spectrum.

By using $W_L = \rho V_L$ (see figure 1), we replace this system of equations with the following system

$$\frac{\eta}{2}\ddot{V_k} + \dot{V_k} = 2\rho\sum_n [(1-\rho\bar{f}_n)(V_k\bar{f}_n - \frac{\partial \bar{f}_n}{\partial V_k})] \quad \forall k < L. \tag{11}$$

$$\frac{\eta}{2}\ddot{W}_L + \dot{W}_L = 2\sum_n [(1-\overline{g}_n)\frac{\partial \overline{g}_n}{\partial W_L})] - 2\lambda W_L. \tag{12}$$

As a sanity check we see that $W^T\dot{W} = \rho\dot{\rho}$ since $W_L = \rho_L V_L$, $\|V_L\| = 1$, $g_n = \rho f_n$.

We multiply the last equation on the left by W^T obtaining

$$\frac{\eta}{2}W^T\ddot{W}_L + W^T\dot{W}_L = 2\sum_n (1-\overline{g}_n)\overline{g}_n - 2\lambda W_L^2 \tag{13}$$

We change variables in the last equation:

$$\frac{\eta}{2}W^T\ddot{W}_L + W^T\dot{W}_L = 2\rho\sum_n [(1-\rho\bar{f}_n)\bar{f}_n) - 2\lambda\rho^2 \tag{14}$$

Since $\dot{W}_L = \rho\dot{V}_L + \dot{\rho}V_L$ and $\ddot{W}_L = 2\dot{\rho}\dot{V}_L + \rho\ddot{V}_L + \ddot{\rho}V_L$, the last equation becomes for $\dot{\rho} = 0$

$$\rho V^T\frac{\eta}{2}\rho\ddot{V}_L + \rho V^T\rho\dot{V}_L = 2\rho\sum_n [(1-\rho\bar{f}_n)\bar{f}_n) - 2\lambda\rho^2 \tag{15}$$

Now notice the following simple relation between accelerations and velocities: $\frac{\partial(V^TV)}{\partial t} = 2V^T\dot{V}$ and $\frac{\partial^2(V^TV)}{\partial t^2} = 2(\dot{V}^T\dot{V} + V^T\ddot{V}) = 2(\|\dot{V}\|^2 + V^T\ddot{V})$. If the norm $\|V_L\|$ is constant, then $\frac{\partial^2(V^TV)}{\partial t^2} = 0$. It follows $V^T\ddot{V} = -\|\dot{V}\|^2$.

Thus

$$\rho^2 V^T\frac{\eta}{2}\ddot{V}_L = 2\rho\sum_n ((1-\rho\bar{f}_n)\bar{f}_n - 2\lambda\rho^2 \tag{16}$$

$$-\frac{\eta}{2}\rho\|\dot{V}_L\|^2 = 2\sum_n (1-\rho\bar{f}_n)\bar{f}_n - 2\lambda\rho \tag{17}$$

Equation (9) implies that when $\dot{\rho} = 0$ then $\|\dot{V}_L\|^2 = 0$.

6 The origin of SGD noise and a bias towards low-rank weight matrices

An intriguing argument for small rank weight matrices is the following observation that follows from Equation (5) (see also [8]).

Lemma 4 *Suppose that there is SGD convergence – that is* $\dot{\rho} = 0, \quad \dot{V_k} = 0, \quad \forall S$ *where* S *is a minibatch – for* $\lambda > 0$*. Then the matrices* V_k *have rank* 1*.*

Proof

The condition

$$V_k f_n = \frac{\partial f_n}{\partial V_k}, \quad \forall k = 1, \cdots, L, \quad \forall n \tag{18}$$

represents a strong set of constraints on the weights of the network. Suppose

$$f(x) = (V_L \sigma(V_{L-1} \cdots \sigma(V_1 x))) \tag{19}$$

where $\sigma(x) = \sigma'(x)x$. This equation can be rewritten for each training example as

$$f(x_j) = V_L D_{L-1}(x_j) V_{L-1} \cdots\cdots V_{k+1} D_k(x_j) V_k \cdots D_1(x_j) V_1 x_j \tag{20}$$

where $D_k(x_j)$ is a diagonal matrix with 0 and 1 entries depending on whether the corresponding ReLU is active or not for the specific input x_j, that is $D_{k-1}(x_j) = diag[\sigma'(N_k(x_j)]$ with $N_k(x_j)$ the input to layer k.

Call $V_L D_{L-1}(x) V_{L-1} \cdots V_{k+1} D_k(x) = a^T$ and $D_{k-1}(x) V_{k-1} D_{k-2}(x) \cdots D_1(x) V_1 x = b$. Then $f(x) = a^T V_k b$ and $\frac{\partial f}{\partial V_k} = ab^T$[8]. As sanity checks, $f^T = b^T V_k^T a = f$; furthermore, the structural lemma Equation (49) gives

$$\sum_{i,j} \frac{\partial f(V;x)}{\partial V_k^{i,j}} V_k^{i,j} = \sum_{i,j} a^i b^j V_k^{i,j} = f(x). \tag{21}$$

Then, Equation (18) becomes

$$V_k f = [V_L D_{L-1}(x) V_{L-1} \cdots V_{k+1} D_k(x)]^T D_{k-1}(x) V_{k-1} D_{k-2}(x) \cdots D_1(x) V_1 x = ab^T. \tag{22}$$

Thus, V_k is a rank one matrix. ■

More in general consider a multi-output deep network, where f is then a C-dimensional vector. Then $f = AV_k b$ and $\frac{\partial f_n}{\partial V_k} = AJb$, where J is the matrix with element $J_{i,j} = 1$. This means that $\frac{\partial f_n}{\partial V_k} = A_{c,i} b_j$. Thus for each output component the constraint of rank one applies.

As a consequence, Equation 5, in the case of SGD, becomes, for asymptotic equilibrium across the mini-batches of size N_b

$$\dot{V}_k = \frac{2}{N_b} \rho \sum_n \left[(1 - \rho \bar{f}_n) \left(-V_k \bar{f}_n + \frac{\partial \bar{f}_n}{\partial V_k} \right) \right]. \tag{23}$$

The equation implies that *at SGD equilibrium $\dot{V}_k = 0$, the V_k are linear combinations of N_b rank one matrices*. Note that among the N_b terms, those corresponding to interpolation have zero weight.

6.1 Origin of SGD noise

Lemma 4 says that there cannot be convergence to a unique set of V_k that satisfy equilibrium for all minibatches. More details of the argument are illustrated in [41]. When $\lambda = 0$, interpolation of all data points is expected: in this case, the GD equilibrium can be reached without any constraint on the V_k. This is also the situation in which the SGD noise is expected to essentially disappear: compare the histograms on the left and the right hand side of Figure 7.

Thus, during training, the solution V_k is not the same for all n: there is *no convergence to a unique solution* but instead attempts to "jump" from one to another during training.

The absence of convergence to a unique solution is not surprising for SGD when the landscape is not convex[9] [10].

[8] When $f = a^T b$, then $\frac{\partial f}{\partial a} = b$ and $\frac{\partial f}{\partial b} = a$

[9] Suppose that the loss landscape has two equivalent minima separated by a low, smooth hill. SGD, as well as as a standard Langevin process, may jump between the two minima, while the average of the SGD parameters may correspond to the hill in between!

[10] The following theorem ([42]), that says that the set of full rank matrices in $\mathbf{R}^{m,n}$ is connected, may be relevant, since it implies that matrices of the same rank are on a smooth manifold while matrices of a different rank are on disconnected manifolds.

Theorem 3 *The set of all $m \times n$ rectangular real matrices of rank r has only one connected component when $m \neq n$ or $r < m = n$. Furthermore, all these connected components are connected by analytic regular arcs.*

6.2 Margins variance with and without regularization

Unregularized square loss minimization in which we have interpolation of all labels implies that all margins are the same ($\sigma^2 = \sum_n (\bar{f}_n - \sum \bar{f}_n)^2) = 0$). We have shown however that in SGD, differently from GD, there may be SGD noise. This is equivalent to saying that that SGD does not converge to a unique solution that corresponds to zero gradient for all data point.

The arguments in the previous section implies that there will be variance in the values of f across different minibatches and correspondingly for the associated values of ρ. Appendix D argues that the variance of the noise in the margins should depend on

We start by considering $\mathcal{L}$ in a rather ideal case: for a specific minibatch of size B $\lambda > 0$ the network reaches the same margins $\bar{f}_n = \frac{1}{\rho_{eq}}$ when n is in the minibatch,

as with $\lambda = 0$ (it is the same network), but of course $\rho = \frac{\mu}{B+\lambda}$ is now smaller than in the case $\lambda = 0$. A realistic alternative is that different V_k for different minibatches allow *the network to reach margin* $\bar{f}_n = \frac{1}{\rho_0}$ *for* M *points and higher margin for the remaining* $N - M$ *points, allowing interpolation.* There are of course other situations with $\sigma > 0$ but we focus on the latter one because configurations with interpolating points plus other points with smaller ρ have smaller $\mathcal{L}$ than other configurations[11]. Of course this correspond to the situation in which there is no single set of V_k for all training data, that is $\dot{V}_k$ is not zero for all n, that is there is no common equilibrium for all minibatches in SGD. The following lemma shows that the minimal value of the loss is achieved when M samples are interpolated.

Lemma 5 *For* $\lambda > 0$ *the network that ideally were to achieve* $\frac{1}{\rho_0}$ *margin for all points, would yield a solution with norm* $\rho_\lambda = \frac{\rho_0}{1+\rho_0^2 \frac{\lambda}{N}}$*, corresponding to quasi-interpolation, with a gap to interpolation which is* $\epsilon = 1 - \rho_0 \frac{1}{\rho_\lambda} = \lambda \rho_0$*. If there exists a set of weights providing interpolation for all but* M *data points, the margins will not be all equal but will still be within* $\lambda \frac{\rho_0}{M}$ *of one another.*

Notice that the gap to interpolation, called ϵ in the lemma, is the same ϵ which is relevant for Neural Collapse (see later assumption 1). If, in this binary case, we consider margins separately for the $+$ and $-$ one-hot encodings, as in the multiclass case (see later), then $1 > f^+(x_+) > 1 - \epsilon$ and $\epsilon > f^+(x_-) > 0$, with $\epsilon = \lambda \frac{\rho_0}{M}$ and fluctuations around it (for individual data points) of the order of $\delta\epsilon = \lambda \frac{\rho_0}{M}$. The proof of the lemma is in Appendix D. Figure 7 shows that the variance of the noise as predicted depends on λ and becomes very small when $\lambda = 0$ (for the square loss).

6.2.1 Rank one Constraint for Exponential-Type Loss Functions

Assume that a deep multilayer neural network $f(x) = W^L \sigma(W^{L-1} \dots \sigma(W^1 x))$ is trained on a binary classification task with weight normalization with respect to an exponential loss function without weight decay. The stationary points of the SGD flow of the normalized weight matrices V_k are given (see [8, 3]) for any finite $\rho = \prod_{k=1}^L \rho_k$, where $\rho_k = \|W_k\|_2$ by

$$\sum_n^B e^{-\rho y_n \hat{f}_n} y_n \left(\frac{\partial \hat{f}_n}{\partial V_k} - V_k \hat{f}_n \right) = 0, \tag{24}$$

where $\hat{f}_n$ is the scalar output of the ReLU network with normalized weight matrices when the input is x_n, y_n is the corresponding binary label and B is the size of the minibatches. For any large but finite value of ρ, the equation shows a similar rank one constraint on the V_k weight matrices, *even with* $\lambda = 0$. Figure shows that indeed, unlike the square loss case, there is noise in the margin, independently of the presence or absence of regularization, as predicted by our analysis.

Figure 8 shows that as predicted there is SGD noise even for $\lambda = 0$ for the cross-entropy loss in the presence of normalization.

6.3 Remarks

- *CNNs are excluded.* Notice that Toeplitz matrices cannot have rank one, apart from rather special cases. The low rank argument does not apply to convolutional networks in any case because

[11] For instance consider a configuration in which $N - M$ points can reach margins larger than $\frac{1}{\rho_0}$, but the remaining M points have lower margins than $\frac{1}{\rho_0}$ resulting in a higher square loss).

the convolution is assumed before training and thus equation 3 does not apply as it stands. Convolutional networks have a restricted number of relevant parameters but for reasons different from the low rank argument.

- *ResNets* with skip connections from every layer to the output may be ideal for peeling out the relevant vector of parameters, each one corresponding to a higher degree of nonlinearity in the network.

- *Intuition:* A common intuition for why there should be a bias towards low rank is that the network is trying to reach the maximum $y_n f_n$ for each training example n with the minimum ρ. This suggests *an implicit minimization of the "stable rank"*, defined as the ratio of the square of the Frobenius norm of each weight matrix and the square of its spectral norm $\frac{||V||_F^2}{||V||^2} = \frac{\sum_j \sigma_j^2}{\max_i \sigma_i^2}$, as described in [11]. This argument, however, is probably a separate mechanism wrt the low rank bias of SGD described earlier.

- *LM normalization is not necessary for square loss.* The previous results were derived assuming normalization by Lagrange multipliers. It turns out [41] that for the square loss normalization is not needed (as it is easy to verify) to yield SGD noise. However, for an exponential type loss, LM normalization can replace regularization wrt to SGD noise and rank constraint. The theoretical prediction is confirmed by experiments (compare Figure 7 and Figure 8).

- Whenever a weight matrix is close to being rank k, the network can be regarded as a composition of a shallow network and a Lipschitz function on $\mathbf{R}^k$.

7 Neural Collapse

A recent paper [4] 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).

Figure 9 shows the phenomenon of NC1 for binary classification: 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, then, the outputs and margins also collapse to the same value. We can see this in the left plot of Figure 7 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 9 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 in fact NC1 never happens with SGD, in the sense that the margins are never, not even asymptotically, exactly equal to each other!

In this section, we show that the phenomenon of neural collapse can be derived from the convergence points of Stochastic Gradient Descent under the square loss with Weight Decay. Unlike section 3, 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 $\mathbb{R}^C$. We train a deep ReLU network $f_W : \mathbb{R}^d \to \mathbb{R}^C$, which takes the form $f_W(x) = W_L \sigma(W_{L-1} \dots W_2 \sigma(W_1 x) \dots)$ with Stochastic Gradient Descent on the square loss with Weight Decay on the parameters of the network. This architecture differs from the one considered in section 3 in that it has C outputs instead of a scalar output. Let the output of the network be $f_W(x) = [f_W^{(1)}(x) \dots f_W^{(C)}(x)]^\top$, and the one-hot target vectors be $y_n = [y_n^{(1)} \dots y_n^{(C)}]^\top$. We will also follow the notation of [4] and use $h : \mathbb{R}^d \to \mathbb{R}^p$ to denote the last layer features of the

deep network. This means that $f_W^{(c)}(x) = \langle W_L^c, h(x)\rangle$. Formally, the loss function takes the form $\mathcal{L}_{\mathcal{S}}(W) = \frac{1}{2NC}\sum_{n=1}^{NC}\sum_{c=1}^{C}\left(y_n^{(c)} - f_W^{(c)}(x_n)\right)^2 + \frac{\lambda}{2}\sum_l \|W_l\|_F^2$. The optimization process iteratively selects a random class-balanced batch $\mathcal{S}' = \cup_{c=1}^{C}\cup_{n=1}^{b}\mathcal{S}_c'$ including b samples $\mathcal{S}_c' \subset \mathcal{S}_c$ per-class and updates the weights of the network is the following manner $W \leftarrow W - \eta\frac{\mathcal{L}_{\mathcal{S}'}(W)}{W}$, 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 W 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 W for which $\frac{\mathcal{L}_{\mathcal{S}'}(W)}{W} = 0$ for all class-balanced batches $\mathcal{S}' \subset \mathcal{S}$.

We make now a key assumption here for the multiclass case, that we conjecture should follow in a weaker form from our analysis of the dynamics. In fact, for binary classification the assumption is Lemma 5 that we proved earlier (the ϵ appearing below is $\epsilon = \lambda\rho_0$).

The assumption is that the solution obtained by Stochastic Gradient Descent satisfies the condition of

Assumption 1 (Symmetric Quasi-interpolation) *Consider a C-class classification problem with inputs in a feature space $\mathcal{X}$ and labels space $\mathbb{R}^C$. A classifier $f_W : \mathcal{X} \to \mathbb{R}^C$ 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} in class c, $f_W^{(c)}(x_{cn}) = 1 - \epsilon$, and $f_W^{(c')}(x_{cn}) = \frac{\epsilon}{C-1}$, where ϵ is the interpolation gap.*

We observe that Lemma 5 shows that a deep network trained with GD under the square loss weakly satisfies the Assumption 1 for the binary case: there is a nonzero variance around the common margin for the x_n inputs. This is an actual gap in our proof below that we hope to correct in future versions. The main result of this section is

Theorem 4 *Let $\mathcal{S} = \cup_{c=1}^{C}\{(x_{cn}, c)\}_{n=1}^{N}$ be a dataset and b be a divisor of N. Let W be an ϵ-optimal point of training a ReLU deep network using SGD with balanced batches of size $B = bC$ on the square loss with weight decay. 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 $M = [\dots\mu_c - \mu_G\dots] \in \mathbb{R}^{C\times p}$. Assume that it satisfies Assumption 2. Then, it also satisfies the conditions for 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: $\frac{W_L}{\|W_L\|_F} = \frac{M^\top}{\|M|_F}$;*
- *NC4: $\arg\max_{c\in[C]} f_W^c(x) = \arg\min_{c\in[C]} \|\mu_c - h(x)\|$.*

Proof Our training objective is $\mathcal{L}_{\mathcal{S}}(W) = \frac{1}{2NC}\sum_{n=1}^{NC}\sum_{c=1}^{C}\left(y_n^{(c)} - f_W^{(c)}(x_n)\right)^2 + \frac{\lambda}{2}\sum_l \|W_l\|_F^2$. We use SGD with balanced batches to train the network. Each step taken by SGD takes the form $-\eta\frac{\partial\mathcal{L}_{\mathcal{S}'}}{\partial W}$, 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 $\frac{\partial\mathcal{L}_{\mathcal{S}'}}{\partial W} = 0$ 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. Let us analyze the dynamics of the last layer, considering each classifier vector W_L^c of W_L, separately:

$$
\begin{aligned}
0 = \frac{\partial\mathcal{L}_{\mathcal{S}'}}{\partial W_L^c} &= \frac{1}{B}\sum_{c'=1}^{C}\sum_{n=1}^{b}\left(\hat{y}_{c'n}^c - \langle W_L^c, h(\hat{x}_{c'n})\rangle\right)\cdot h(\hat{x}_{c'n}) - \lambda W_L^c \\
&= \frac{1}{B}\sum_{n=1}^{b}\left(1 - \langle W_L^c, h(\hat{x}_{cn})\rangle\right)\cdot h(\hat{x}_{cn}) \\
&\quad + \frac{1}{B}\sum_{c'\in[C]\setminus\{c\}}\sum_{n=1}^{b}\left(-\langle W_L^c, h(\hat{x}_{c'n})\rangle\right)\cdot h(\hat{x}_{c',n}) - \lambda W_L^c.
\end{aligned}
\tag{25}
$$

Let us consider solutions that achieve *symmetric quasi-interpolation*, with $f_W^{(c)}(\hat{x}_{cn}) = 1 - \epsilon$, and $f_W^{(c)}(\hat{x}_{c'n}) = \frac{\epsilon}{C-1}$. Hence, we have

$$\frac{1}{B}\sum_{n=1}^{b}\epsilon h(\hat{x}_{cn}) - \frac{1}{B}\sum_{c'\in[C]\setminus\{c\}}\sum_{n=1}^{b}\frac{\epsilon}{C-1}h(\hat{x}_{c'n}) - \lambda W_L^c = 0. \tag{26}$$

In addition, consider a second batch $\mathcal{S}''$ that differs from $\mathcal{S}'$ by only one sample $\hat{x}'_{cn}$ instead of $\hat{x}_{cn}$ from class c. By applying the previous Eq. (26) for $\mathcal{S}'$ and for $\mathcal{S}''$, we can obtain $h(\hat{x}_{cn}) = h(\hat{x}'_{cn})$, 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

$$\begin{aligned}
0 &= \frac{1}{k}\sum_{i=1}^{k}\frac{\partial\mathcal{L}_{\mathcal{S}^i}(W)}{\partial W_L^c} \\
&= \frac{\partial\mathcal{L}_{\mathcal{S}}(W)}{\partial W_L^c} \\
&= \frac{1}{NC}\sum_{n}(y_n^c - \langle W_L^c, h(x_n)\rangle)\cdot h(x_n) - \lambda W_L^c \\
&= \frac{1}{NC}\sum_{n=1}^{N}\left(1 - \langle W_L^c, h(x_{cn})\rangle\right)h(x_{cn}) + \frac{1}{NC}\sum_{c'\in[C]\setminus\{c\}}\sum_{n=1}^{N}\left(-\langle W_L^c, h(x_{c'n})\rangle\right)h(x_{c'n}) - \lambda W_L^c.
\end{aligned} \tag{27}$$

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:

$$\frac{\partial\mathcal{L}_{\mathcal{S}}(W)}{\partial W_L^c} = 0 \quad \implies W_L^c = \frac{\epsilon}{\lambda(C-1)}\cdot(\mu_c - \mu_G). \tag{28}$$

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

From Equation (27), we can also see that at convergence with quasi-interpolation, all classifier vectors in the last layer (W_L^c, and hence $\mu_c - \mu_G$) have the same norm:

$$\begin{aligned}
\left\langle W_L^c, \frac{\partial\mathcal{L}_{\mathcal{S}}(W)}{\partial W_L^c}\right\rangle &= \frac{1}{NC}\sum_{n}\left(y_n^c - f_W^{(c)}(x_n)\right)\cdot f_W^{(c)}(x_n) - \lambda\|W_L^c\|_2^2 = 0 \\
\implies \|W_L^c\|_2^2 &= \frac{1}{C\lambda}\left(\epsilon - \frac{C}{C-1}\epsilon^2\right)
\end{aligned} \tag{29}$$

From the quasi-interpolation of the correct class label we have that $\langle W_L^c, \mu_c\rangle = 1 - \epsilon$ which means $\langle W_L^c, \mu_G\rangle + \langle W_L^c, \mu_c - \mu_G\rangle = 1 - \epsilon$. Now using Equation (28)

$$\begin{aligned}
\langle W_L^c, \mu_G\rangle &= 1 - \epsilon - \frac{\lambda(C-1)}{\epsilon}\cdot\|W_L^c\|^2 \\
&= 1 - \epsilon - \frac{\lambda(C-1)}{\epsilon}\cdot\frac{1}{C\lambda}\left(\epsilon - \frac{C}{C-1}\epsilon^2\right) = \frac{1}{C}.
\end{aligned} \tag{30}$$

From the quasi-interpolation of the incorrect class labels, we have that $\langle W_L^c, \mu_{c'}\rangle = \frac{\epsilon}{C-1}$, which means that $\langle W_L^c, \mu_{c'} - \mu_G\rangle + \langle W_L^c, \mu_G\rangle = \frac{\epsilon}{C-1}$. Plugging in the previous result and using Equation (29) implies

$$\begin{aligned}
\frac{\lambda(C-1)}{\epsilon}\cdot\langle W_L^c, W_L^{c'}\rangle &= \frac{\epsilon}{C-1} - \frac{1}{C} \\
\implies \langle V_L^c, V_L^{c'}\rangle &= \frac{1}{\|W_L^c\|_2^2}\cdot\frac{\epsilon}{\lambda(C-1)}\cdot\left(\frac{\epsilon}{C-1} - \frac{1}{C}\right) = -\frac{1}{C-1}
\end{aligned} \tag{31}$$

Here $V_L^c = \frac{W_L^c}{\|W_L^c\|_2}$, and we use the fact that all the norms $\|W_L^c\|_2$ are equal. This completes the proof that the normalized classifier parameters form an ETF. Moreover, since $W_L^c \propto \mu_c - \mu_G$ and all the

proportionality constants are independent of c, we obtain $\sum_c W_L^c = 0$. This completes the proof of the NC2 condition. Next, we would like to prove NC4. We consider that our classifier takes the form:

$$\arg\max_{c\in[C]} f_W^c(x) = \arg\max_{c\in[C]} \langle W_L^c, h(x)\rangle = \arg\max_{c\in[C]} \alpha\langle \mu_c - \mu_G, h\rangle, \tag{32}$$

We subtract and add μ_G to obtain an equivalent classifier of the form:

$$\arg\max_{c\in[C]} \left(\alpha\langle \mu_c - \mu_G, h(x) - \mu_G\rangle + \alpha\langle \mu_c - \mu_G, \mu_G\rangle\right). \tag{33}$$

Notice that because of NC2 (equiangularity) $\langle \mu_c, \mu_G\rangle$ is the same across all $c \in [C]$, so $\alpha\langle \mu_c - \mu_G, \mu_G\rangle$ is simply a constant. Hence, the classifier is equivalent to $\arg\max_{c\in[C]}\langle \mu_c - \mu_G, h(x) - \mu_G\rangle$, which is also equivalent to $\arg\min_{c\in[C]} \|\mu_c - h(x)\|$, as desired. ■

It is of interest to note here that in this quasi-interpolation setting, the functional classification margin is given by $\eta_n = f_{y_n} - \max_{c\neq y_n} f_c = 1 - \epsilon - \frac{\epsilon}{C-1} = 1 - \frac{C}{C-1}\epsilon$. The larger the margin, the smaller is ϵ. Equation (29) shows that the norms of the classifier weights are given by $\|W_L^c\|_2^2 = \frac{N\epsilon}{\lambda}\eta$. As we mentioned in Lemma 5, for a non-zero value of λ we expect some small interpolation gap ϵ. In the binary case this is given by $\epsilon = \lambda\rho_0$. Plugging this relationship into Equation (29) we obtain $\|W_c\|^2 \approx \frac{N(1-\epsilon)}{\sum_n |f_n|^2}\eta$. This means that the lengths of the classifier simplex ETF are proportional to the margin.

As explained, in order to prove Neural Collapse, we needed the very strong Assumption 2, which is stricter than the bounds obtained from an analysis of the dynamics (see Lemma 5). Can we relax the assumption of equal margins and do better? Below we consider such a relaxation, in which we allow some variance in the margins, and predict the impact of that onto the variance of the hidden features h.

Theorem 5 *Let $\mathcal{S} = \cup_{c=1}^C \{(x_{cn}, c)\}_{n=1}^N$ be a dataset. Let W be a vector of parameters. Let $\mu_c = \frac{1}{N}\sum_{n=1}^N h(x_{cn})$. Then, we have the following bounds*

$$\frac{\mathbb{E}_c\left[\mathbb{E}_n[\|h(x_{cn}) - \mu_c\|]^2\right]}{\max_c \mathbb{E}_m\left[\|h(x_{cm})\|^2\right]} \leq \mathbb{E}_{c,n}\left[\frac{4}{(1 - f_W^c(x_{cn}))^2}\right] \cdot \left(2\max_c \mathrm{Var}_n(f_W^c(x_{cn})) + \frac{\max_{c,S'} \|\frac{\partial \mathcal{L}_{S'}(W)}{\partial W_L^c}\|^2}{\max_c \mathbb{E}_m[\|h(x_{cm})\|^2]}\right),$$

where $S' \subset S$ is a balanced batch of size $B = bC$.

Proof Our training objective is $\mathcal{L}_{\mathcal{S}}(W) = \frac{1}{2NC}\sum_{n=1}^{NC}\sum_{c=1}^{C}\left(y_n^{(c)} - f_W^{(c)}(x_n)\right)^2 + \frac{\lambda}{2}\sum_l \|W_l\|_F^2$. We use SGD with balanced batches to train the network. Each step taken by SGD takes the form $-\eta\frac{\partial \mathcal{L}_{\mathcal{S}'}}{\partial W}$, where $\mathcal{S}' \subset \mathcal{S}$ is a balanced batch containing exactly b samples per class. Let $\mathcal{S}' = \cup_{c=1}^C \cup_{n=1}^b \{(\hat{x}_{cn}, \hat{y}_{cn})\}$ and $\mathcal{S}'' = \cup_{c=1}^C \cup_{n=1}^b \{(\hat{x}'_{cn}, \hat{y}'_{cn})\}$ be two balanced batches that differ by only one sample $\hat{x}'_{ci}$ instead of $\hat{x}_{ci}$. Let us analyze the dynamics of the last layer, considering each classifier vector W_L^c of W_L separately:

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\mathcal{S}'}}{\partial W_L^c} &= \frac{1}{B}\sum_{c'=1}^{C}\sum_{n=1}^{b}(\hat{y}_{c'n}^c - \langle W_L^c, h(\hat{x}_{c'n})\rangle)\cdot h(\hat{x}_{c'n}) - \lambda W_L^c \\
&= \frac{1}{B}\sum_{n=1}^{b}(1 - \langle W_L^c, h(\hat{x}_{cn})\rangle)\cdot h(\hat{x}_{cn}) \\
&\quad + \frac{1}{B}\sum_{c'\in[C]\setminus\{c\}}\sum_{n=1}^{b}(-\langle W_L^c, h(\hat{x}_{c'n})\rangle)\cdot h(\hat{x}_{c',n}) - \lambda W_L^c
\end{aligned}$$

By subtracting the previous equation when applied once for $\mathcal{S}'$ and once for $\mathcal{S}''$, we obtain the following identity

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\mathcal{S}'}}{\partial W_L^c} - \frac{\partial \mathcal{L}_{\mathcal{S}''}}{\partial W_L^c} &= (1 - \langle W_L^c, h(\hat{x}_{cn})\rangle)\cdot h(\hat{x}_{cn}) - (1 - \langle W_L^c, h(\hat{x}_{cn})\rangle)\cdot h(\hat{x}'_{cn}) \\
&= (1 - f_W^c(\hat{x}_{cn}))\cdot h(\hat{x}_{cn}) - (1 - f_W^c(\hat{x}'_{cn}))\cdot h(\hat{x}'_{cn})
\end{aligned} \tag{34}$$

In particular,

$$2\epsilon \geq \left\| \frac{\partial \mathcal{L}_{\mathcal{S}'}}{\partial W_L^c} \right\| + \left\| \frac{\partial \mathcal{L}_{\mathcal{S}''}}{\partial W_L^c} \right\| \geq \|(1 - f_W^c(\hat{x}_{cn})) \cdot h(\hat{x}_{cn}) - (1 - f_W^c(\hat{x}'_{cn})) \cdot h(\hat{x}'_{cn})\|, \tag{35}$$

where $\epsilon := \max_{c,\mathcal{S}'} \|\frac{\partial \mathcal{L}_{\mathcal{S}'}(W)}{\partial W_L^c}\|$. Let x_{cn} and x_{cm} be two samples from the same class c. Assume that x_{cn} and x_{cm} are distributed independently and uniformly over $\{x \mid (x, c) \in \mathcal{S}_c\}$ and $c \sim U[C]$. Let $\epsilon_{cn} = 1 - f_W^c(x_{cn})$. We can rewrite Equation (35) as

$$\|\epsilon_{cn} h(x_{cn}) - \epsilon_{cm} h(x_{cm})\| \leq 2\epsilon \tag{36}$$

Therefore, by denoting $\Delta = \epsilon_{cn} - \epsilon_{cm}$, by Equation (26) we obtain that

$$\|\epsilon_{cn}(h(x_{cn}) - h(x_{cm}))\| \leq |\Delta| \cdot \|h(x_{cm})\| + 2\epsilon. \tag{37}$$

In particular,

$$\begin{aligned} |\epsilon_{cn}| \cdot \|h(x_{cn}) - h(x_{cm})\| &\leq |\Delta| \cdot \|h(x_{cm})\| + 2\epsilon \\ &\leq (|\epsilon_{cn} - \mathbb{E}[\epsilon_{cn}]| + |\epsilon_{cm} - \mathbb{E}[\epsilon_{cm}]|) \cdot (\|h(x_{cm})\|) + 2\epsilon \\ &= (\Delta_1 + \Delta_2) \cdot \|h(x_{cm})\| + 2\epsilon, \end{aligned} \tag{38}$$

where $\Delta_1 := |\epsilon_{cn} - \mathbb{E}[\epsilon_{cn}]|$ and $\Delta_2 := |\epsilon_{cm} - \mathbb{E}[\epsilon_{cm}]|$. Hence,

$$\begin{aligned} &\mathbb{E}_{n,m}[\|h(x_{cn}) - h(x_{cm})\|]^2 \\ &\leq \mathbb{E}_{n,m}\left[\frac{(\Delta_1 + \Delta_2) \cdot \|h(x_{cm})\|}{|\epsilon_{cn}|} + \frac{2\epsilon}{|\epsilon_{cn}|}\right]^2 \\ &\leq 2\mathbb{E}_{n,m}\left[\frac{(\Delta_1 + \Delta_2) \cdot \|h(x_{cm})\|}{|\epsilon_{cn}|}\right]^2 + 4\epsilon \cdot \mathbb{E}_n\left[|\epsilon_{cn}|^{-1}\right]^2 \\ &\leq 2\mathbb{E}_{n,m}\left[|\epsilon_{cn}|^{-2} \cdot \|h(x_{cm})\|^2\right] \cdot \mathbb{E}_{n,m}\left[(\Delta_1 + \Delta_2)^2\right] + 4\epsilon^2 \cdot \mathbb{E}_n\left[|\epsilon_{cn}|^{-1}\right]^2 \\ &= 2\mathbb{E}_n\left[|\epsilon_{cn}|^{-2}\right] \cdot \mathbb{E}_m[\|h(x_{cm})\|^2] \cdot \mathbb{E}_{n,m}\left[(\Delta_1 + \Delta_2)^2\right] + 4\epsilon^2 \cdot \mathbb{E}_n\left[|\epsilon_{cn}|^{-1}\right]^2 \\ &\leq 2\mathbb{E}_n[|\epsilon_{cn}|^{-2}] \cdot \mathbb{E}_m\left[\|h(x_{cm})\|^2\right] \cdot \mathbb{E}_{n,m}\left[(\Delta_1 + \Delta_2)^2\right] + 4\epsilon^2 \cdot \mathbb{E}_n\left[|\epsilon_{cn}|^{-2}\right] \\ &= 2\mathbb{E}_n[\epsilon_{cn}^{-2}] \cdot \mathbb{E}_m\left[\|h(x_{cm})\|^2\right] \cdot \mathbb{E}_{n,m}\left[(\Delta_1 + \Delta_2)^2\right] + 4\epsilon^2 \cdot \mathbb{E}_n\left[\epsilon_{cn}^{-2}\right] \end{aligned}$$

where the expectations are taken with respect to $c \sim U[C]$ and $n, m \sim U[N]$ and the third inequality follows from the Cauchy-Schwarz inequality and the last one follows from Jensen's inequality. In addition, for all $c \in [C]$,

$$\mathbb{E}_n[\|h(x_{cn}) - \mu_c\|] \leq \mathbb{E}_{n,m}[\|h(x_{cn}) - h(x_{cm})\|]. \tag{39}$$

In particular,

$$\mathbb{E}_c[\mathbb{E}_n[\|h(x_{cn}) - \mu_c\|]^2] \leq \mathbb{E}_c[\mathbb{E}_{n,m}[\|h(x_{cn}) - h(x_{cm})\|]^2]. \tag{40}$$

Hence, we have

$$\begin{aligned} \mathbb{E}_c\left[\mathbb{E}_n[\|h(x_{cn}) - \mu_c\|]^2\right] &\leq 2\mathbb{E}_c\left[\mathbb{E}_n[\epsilon_{cn}^{-2}] \cdot \mathbb{E}_m\left[\|h(x_{cm})\|^2\right] \cdot \mathbb{E}_{n,m}\left[(\Delta_1 + \Delta_2)^2\right]\right] + 4\epsilon \cdot \mathbb{E}_n\left[\epsilon_{cn}^{-2}\right] \\ &\leq 2\mathbb{E}_{c,n}[\epsilon_{cn}^{-2}] \cdot \max_c \mathbb{E}_m\left[\|h(x_{cm})\|^2\right] \cdot \max_c \mathbb{E}_{n,m}\left[(\Delta_1 + \Delta_2)^2\right] + 4\epsilon \cdot \mathbb{E}_n\left[\epsilon_{cn}^{-2}\right] \\ &\leq 4\mathbb{E}_{c,n}[\epsilon_{cn}^{-2}] \cdot \max_c \mathbb{E}_m\left[\|h(x_{cm})\|^2\right] \cdot \max_c \mathbb{E}_{n,m}\left[\Delta_1^2 + \Delta_2^2\right] + 4\epsilon \cdot \mathbb{E}_n\left[\epsilon_{cn}^{-1}\right]^2 \\ &\leq 8\mathbb{E}_{c,n}[\epsilon_{cn}^{-2}] \cdot \max_c \mathbb{E}_m\left[\|h(x_{cm})\|^2\right] \cdot \max_c \mathbb{E}_n\left[\Delta_1^2\right] + 4\epsilon \cdot \mathbb{E}_n\left[\epsilon_{cn}^{-2}\right] \end{aligned} \tag{41}$$

which finally gives the desired inequality,

$$\frac{\mathbb{E}_c\left[\mathbb{E}_n[\|h(x_{cn}) - \mu_c\|]^2\right]}{\max_c \mathbb{E}_m\left[\|h(x_{cm})\|^2\right]} \leq \mathbb{E}_{c,n}\left[\frac{4}{(1 - f_W^c(x_{cn}))^2}\right] \cdot \left(2 \max_c \mathrm{Var}_n(f_W^c(x_{cn})) + \frac{\epsilon^2}{\max_c \mathbb{E}_m[\|h(x_{cm})\|^2]}\right). \tag{42}$$

■

The dependence of the denominator in the bound on $\min_{(x,y)\in\mathcal{S}}(1 - f_W^y(x))^2$ is potentially worrisome here, if there are datapoints that can be exactly interpolated in presence of non-zero weight decay. If some of the datapoints interpolate exactly, we cannot guarantee that their features collapse – excluding them from the analysis however, we still get a guarantee of NC for all the other datapoints.

Other settings A similar version of the above proof can be adapted to the case of Stochastic Gradient Descent (SGD), where we can show that the NC conditions are met in expectation. We also show in section 5 of the Supplementary Material that an extension of this proof technique to the exponential loss case (a proxy for cross-entropy loss) requires small batch SGD to achieve an approximation of the NC1 property.

Remarks

- To achieve Neural Collapse, we cannot exactly interpolate the labels in homogenous neural networks. If we fit the data, Neural Collapse does not happen.
- The analysis of the loss landscape and of the qualitative dynamics under the square loss in section 4.3 and in section 4.2 implies that all quasi-interpolating solutions with $\rho \geq \rho_0$ and $\lambda > 0$ yield neural collapse and have its four properties, including the ETF property.
- SGD is required in our proof of NC1;
- Our proof uses Weight Decay – that turns out to be necessary – for neural collapse (NC1 to NC4) under the square loss, and can be extended to the case of normalization;
- The length of the vectors in the Simplex ETF that defines the classifier is proportional to the training margin;
- 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 (that is small ρ);
- In particular the analysis above predicts Neural Collapse for randomly labeled CIFAR10.

8 Generalization

Assume that the square loss is exactly zero and the margins $\bar{f}_n$ are all the same. Then recall simple generalization bounds that hold with probability at least $(1-\delta)$, $\forall g \in \mathbb{G}$ of the form [43]:

$$|L(g) - \hat{L}(g)| \leq 2\mathbb{R}_N(\mathbb{G}) + c_2\sqrt{\frac{\ln(\frac{1}{\delta})}{2N}} \tag{43}$$

where $L(g) = \mathbf{E}[\ell_\gamma(g(x), y)]$ is the expected loss, $\hat{L}(g)$ is the empirical loss, $\mathbb{R}_N(\mathbb{G})$ is the empirical Rademacher average of the class of functions $\mathbb{G}$ measuring its complexity; c_2 reflects the Lipschitz constant of the loss function and the architecture of the network. The loss function here is the *ramp loss* $\ell_\gamma(g(x), y)$ defined as

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

We define $\ell_{\gamma=0}(y, y')$ as the standard $0-1$ classification error and observe that $\ell_{\gamma=0}(y, y') < \ell_{\gamma>0}(y, y')$.

We now use the observation that, because of homogeneity of the ReLU networks, the empirical Rademacher complexity satisfies the property,

$$\mathbb{R}_N(\mathbb{G}) = \rho\mathbb{R}_N(\mathbb{F}), \tag{44}$$

where $\mathbb{G}$ is the space of functions of our unnormalized networks and $\mathbb{F}$ denotes the corresponding normalized networks. This yields

$$|L(g) - \hat{L}(g)| \leq 2\rho\mathbb{R}_N(\mathbb{F}) + c_2\sqrt{\frac{\ln(\frac{1}{\delta})}{2N}}. \tag{45}$$

Furthermore it is clear because of homogeneity of RELU and bounds on norms of product of matrices that

$$\mathbb{R}_N(\mathbb{F}) \leq \frac{1}{\sqrt{(N)}}. \tag{46}$$

It is known that the Rademacher complexity of normalized deep ReLU networks with L layers $\mathbb{R}_N(\mathbb{F})$ behaves [44] as $\mathbb{R}_N(\mathbb{F}) = \mathcal{O}\sqrt{(\frac{L}{N})}$, where the term $\sqrt{L}$ is effectively included in our ρ, which is the product of the L ρ_k. Furthermore under certain conditions [12] faster rates of convergence – up to $\mathbb{R}_N(\mathbb{F}) = \mathcal{O}(\frac{\sqrt{L}}{N})$ – may be possible [45]. A rate between $\frac{1}{\sqrt{N}}$ and $\frac{1}{N}$ but close to $\frac{1}{N}$ would imply that the bound given by Equation (43) is non-vacuous, since $\rho << N$ in the experiments described in this paper with LM (see, for instance, Figures 6 and 7). In fact Figure 11 suggests that the rate of convergence is relatively "fast" and therefore that the margin bound is non-vacuous! Furthermore, the use of BN yields a value of ρ around 3 as shown in Figure 22 which means that the bound on the Rademacher complexity is much smaller than $\sqrt{(N)}$, even when N is just $N = 500$. See Figure 12.

It is important to remark that it is impossible to compare directly experiments with ρ obtained with $\lambda = 0$ and with ρ obtained with $\lambda > 0$. The reason is that in the case of $\lambda = 0$ we can expect to find solutions to $\rho\overline{f_n} = 1$, for most values of ρ, given sufficient overparametrization. Without regularization (and especially if normalization is used) the solution will be found close to the ρ associated with initialization. An example is shown in 25. In this situation, similar to linear regression of random features, it seems likely that optimization mainly acts on the weights of the last layer. Alternatively, this is a case in which the bound should be better estimated in terms of the norm $||W_k - A_k||$ where the A_k are appropriate reference matrices [46]: here the could be chosen to be the weight matrices at initialization instead of zero matrices.

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 use Equation 45 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

$$L_0(f^a) \leq A\rho_a + \epsilon \tag{47}$$

and

$$L_0(f^b) \leq A\rho_b + \epsilon \tag{48}$$

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. Absolute bounds on generalization based on minimum margin (see [34]) are often loose.

Figure 13 shows clearly that increasing the percentage of random labels increases ρ needed to maintain interpolation – decreasing the margin – and that at the same time the test error increases as expected from Equation 45. This monotonic relation between margin and accuracy at test seems to break down for small differences in margin as shown in Figure 14, though the significance of the effect is unclear. Of course this kind of behavior is not inconsistent with an upper bound.

[12] Such a condition is the existence of a function in the hypothesis space which is consistent with the data, implying $\hat{L}(g) = 0$; another is smoothness of the loss function. The first one is satisfied in our case; the second one can be satisfied by replacing $\ell_\gamma(y, y')$ above with a smoothed version of it (see Theorem 6 in [45]).

9 Discussion

Our analysis of the dynamics of gradient flow and gradient descent shows that we expect quasi-interpolating solutions with small ρ in the presence of weight decay. Similar small ρ solutions may be occasionally be obtained with careful, small intialization triggering an increase of ρ from about zero during training. More frequently, with sizable initialization training without weight decay will quickly converge to solutions that are likely to be NTK-like.

With SGD and weight decay we uncover a bias of convergence towards not only small ρ but also small rank solutions. At the same time, we show that SGD with regularization yields a unavoidable noise, which, strictly speaking makes exact convergence impossible, even asymptotically. However, within a small ϵ margins do converge to each other implying that the first condition for neural collapse is satisfied. We also show that the other conditions are also satisfied.

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[13]. In fact, our prediction of Neural Collapse for randomly labeled CIFAR10, was confirmed originally in preliminary experiments by our collaborators (Papyan et al.) and more recently in other papers (see for instance, [47]). Interestingly, there is a connection between Neural Collapse and max margin solutions. Assume there exists a set of weights associated with a quasi-interpolating solution. Then Corollary 1 says that there exist a continuous connected curve of solutions, with increasing ρ and the same other weights $V_k (k = 1, \cdots, L-1)$, but slightly different V_L that show NC1 but not NC3.

Independently of Neural Collapse, can our analysis of the square loss dynamics and its connection with margin and low rank provide insights on generalization of the solutions of gradient flow? It is well known that large margin is usually associated with good generalization [43]; in the meantime it is also broadly recognized that margin alone does not fully account for generalization in deep nets [34, 48, 49]. Margin in fact provides an upper bound on generalization error, as shown in section 8. 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 inverse of the margin on the training set together with the test error also increases with the percentage of random labels as shown in the supplementary material. However, the upper bound given in section 8 does not explain by itself details of the generalization behavior that we observe for different initializations (see Figure 14), where small differences in margin are actually anticorrelated with small differences in test error. We conjecture that margin together with rank may be sufficient to explain generalization, as discussed briefly in section 8.

The bias towards small ρ solutions induced by regularization for $\lambda > 0$ can be replaced by an implicit bias induced by small initialization and parameters values that allow convergence to the first quasi-interpolating solution for increasing ρ[14] Interestingly, we found that for our networks with weight decay, the bounds on the expected error in terms of Rademacher complexity are not vacuous, since ρ which represents an upper bound on the Rademacher averages of the class of functions

$$\mathbb{R}_N(\mathbb{G})$$

implemented by our deep networks, is smaller than $\sqrt(N)$. This seems to be novel: in our experiments there is overparametrization and thus quasi-interpolation of the training data but also sufficiently small Rademacher complexity to ensure non-vacuous bounds. We will explore the issue in more detail in a forthcoming paper (Xu, Malach and Poggio).

In summary, this paper explains the bias of SGD on the square loss for $\lambda > 0$ towards 1) minimum ρ max margin solutions, and 2) towards small rank of the weight matrices.

[13] Even if margin is likely to be just one of the factors determining out-of-sample performance.

[14] Notice that the generalization bound in Section 4 of the supplementary material does not directly rely on the Weight Decay parameter $\lambda > 0$: there can be generalization without weight decay, depending on the trajectory of the dynamics and thus also on initialization. However, robust convergence to large margins is helped by a non-zero λ, even if λ is quite small. This effect is different from the standard explanation that regularization is needed to force the norm to be small, since a small norm can follow from small initial conditions without regularization. The main effect of $\lambda > 0$ is to eliminate degeneracy of the dynamics at the zero-loss critical points. In fact, $\dot{V}_k = 0$ can be satisfied even when $(V_k f_n - \frac{\partial f_n}{\partial V_k}) \neq 0$, implying that any interpolating solution can satisfy the equilibrium equations independently of its rank. This degeneracy is expected, since there are infinite sets of ρ and V_k satisfying $\rho V_k = W_k$. The rank constraint may not be effective at the critical points. Setting $\lambda > 0$ avoids this degeneracy.

With respect to the margin 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. Associated with the minimum ρ solutions are upper bounds on the generalization error and properties such as Neural Collapse. Interestingly, these results do not say why deep networks should be better than other classifiers such as kernel machines. We believe that the answer to this question is in approximation properties [50] of certain classes of deep networks and not in any implicit bias of the optimization process – apart from the minimum ρ dynamics described in this paper. In particular, CNN-like networks, in general without weight sharing, enjoy good approximation properties for functions that can be represented in a sparse compositional form. Furthermore, unpublished results (Malach, Poggio) suggest that deep compositional architectures with skip connections also have remarkable optimization properties for a broad subset of the same class of compositional functions.

Acknowledgments 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), and AFSOR-THRL (FA8650-05-C-7262).

References

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

[2] Kaifeng Lyu and Jian Li. Gradient descent maximizes the margin of homogeneous neural networks. *CoRR*, abs/1906.05890, 2019.

[3] Tomaso Poggio, Andrzej Banburski, and Qianli Liao. Theoretical issues in deep networks. *PNAS*, 2020.

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

[5] Mor Shpigel Nacson, Suriya Gunasekar, Jason D. Lee, Nathan Srebro, and Daniel Soudry. Lexicographic and Depth-Sensitive Margins in Homogeneous and Non-Homogeneous Deep Models. *arXiv e-prints*, page arXiv:1905.07325, May 2019.

[6] A. Banburski, Q. Liao, B. Miranda, T. Poggio, L. Rosasco, B. Liang, and J. Hidary. Theory of deep learning III: Dynamics and generalization in deep networks. *CBMM Memo No. 090*, 2019.

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

[8] T. Poggio and Q. Liao. Generalization in deep network classifiers trained with the square loss. *CBMM Memo No. 112*, 2019.

[9] T. Poggio and Y. Cooper. Loss landscape: Sgd has a better view. *CBMM Memo 107*, 2020.

[10] Yaim Cooper. Global minima of overparameterized neural networks. *SIAM Journal on Mathematics of Data Science*, 3(2):676–691, 2021.

[11] Nadav Timor, Gal Vardi, and Ohad Shamir. Implicit regularization towards rank minimization in relu networks. *CoRR*, abs/2201.12760, 2022.

[12] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

[13] Lenaic Chizat and Francis Bach. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. In *Conference on Learning Theory*, pages 1305–1338. PMLR, 2020.

[14] Tengyu Xu, Yi Zhou, Kaiyi Ji, and Yingbin Liang. When will gradient methods converge to max-margin classifier under relu models? *Stat*, 10(1):e354, 2021.

[15] Vidya Muthukumar, Adhyyan Narang, Vignesh Subramanian, Mikhail Belkin, Daniel Hsu, and Anant Sahai. Classification vs regression in overparameterized regimes: Does the loss function matter? *arXiv e-prints*, page arXiv:2005.08054, May 2020.

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

[17] Tengyuan Liang and Benjamin Recht. Interpolating classifiers make few mistakes. *arXiv preprint arXiv:2101.11815*, 2021.

[18] Kai Zhong, Zhao Song, Prateek Jain, Peter L Bartlett, and Inderjit S Dhillon. Recovery guarantees for one-hidden-layer neural networks. In *International conference on machine learning*, pages 4140–4149. PMLR, 2017.

[19] Mahdi Soltanolkotabi, Adel Javanmard, and Jason D Lee. Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *IEEE Transactions on Information Theory*, 65(2):742–769, 2018.

[20] Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019.

[21] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *arXiv preprint arXiv:1812.07956*, 2018.

[22] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*, 2018.

[23] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018.

[24] Zhengdao Chen, Grant M Rotskoff, Joan Bruna, and Eric Vanden-Eijnden. A dynamical central limit theorem for shallow neural networks. *arXiv preprint arXiv:2008.09623*, 2020.

[25] Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332. PMLR, 2019.

[26] Dustin G. Mixon, Hans Parshall, and Jianzong Pi. Neural collapse with unconstrained features. *CoRR*, abs/2011.11619, 2020.

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

[28] Cong Fang, Hangfeng He, Qi Long, and Weijie J. Su. Layer-peeled model: Toward understanding well-trained deep neural networks. *CoRR*, abs/2101.12699, 2021.

[29] Stephan Wojtowytsch et al. On the emergence of tetrahedral symmetry in the final and penultimate layers of neural network classifiers. *arXiv preprint arXiv:2012.05420*, 2020.

[30] Tolga Ergen and Mert Pilanci. Revealing the structure of deep neural networks via convex duality. *arXiv preprint arXiv:2002.09773*, 2020.

[31] 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.

[32] Sanjeev Arora, Zhiyuan Li, and Kaifeng Lyu. Theoretical analysis of auto rate-tuning by batch normalization. *CoRR*, abs/1812.03981, 2018.

[33] Sourav Chatterjee. Convergence of gradient descent for deep neural networks, 2022.

[34] P. Bartlett, D. J. Foster, and M. Telgarsky. Spectrally-normalized margin bounds for neural networks. *ArXiv e-prints*, June 2017.

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

[36] T. Poggio and Y. Cooper. Loss landscape: Sgd can have a better view than gd. *CBMM memo 107*, 2020.

[37] Quynh Nguyen. On connected sublevel sets in deep learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4790–4799. PMLR, 09–15 Jun 2019.

[38] J. Evard and F. Jafari. The set of all rectangular matrices of rank r is connected by analytic arcs. *Proc. of the American Mathematical Society*, pages 413,419, 1994.

[39] David G. T. Barrett and Benoit Dherin. Implicit gradient regularization, 2021.

[40] Daniel Kunin, Javier Sagastuy-Brena, Surya Ganguli, Daniel L. K. Yamins, and Hidenori Tanaka. Neural mechanics: Symmetry and broken conservation laws in deep learning dynamics, 2021.

[41] 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.

[42] J. C. Evard and F. Jafari. The set of all rectangular real matrices is connected by analytic regular arcs. *Proceedings of the American Mathematical Society*, 120(2):413–419, February 1994.

[43] Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. Introduction to statistical learning theory. In *Summer school on machine learning*, pages 169–207. Springer, 2003.

[44] Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-independent sample complexity of neural networks. *CoRR*, abs/1712.06541, 2017.

[45] Nathan Srebro, Karthik Sridharan, and Ambuj Tewari. Smoothness, low-noise and fast rates. *CoRR*, abs/1009.3896, 2010.

[46] Peter L. Bartlett, Dylan J. Foster, and Matus Telgarsky. Spectrally-normalized margin bounds for neural networks. *CoRR*, abs/1706.08498, 2017.

[47] Zhihui Zhu, Tianyu Ding, Jinxin Zhou, Xiao Li, Chong You, Jeremias Sulam, and Qing Qu. A geometric analysis of neural collapse with unconstrained features. *CoRR*, abs/2105.02375, 2021.

[48] Yiding Jiang, Dilip Krishnan, Hossein Mobahi, and Samy Bengio. Predicting the generalization gap in deep networks with margin distributions. *arXiv preprint arXiv:1810.00113*, 2018.

[49] Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*, 2019.

[50] H.N. Mhaskar and T. Poggio. Deep vs. shallow networks: An approximation theory perspective. *Analysis and Applications*, pages 829– 848, 2016.

[51] Tengyuan Liang, Tomaso Poggio, Alexander Rakhlin, and James Stokes. Fisher-rao metric, geometry, and complexity of neural networks. *CoRR*, abs/1711.01530, 2017.

A Structural Lemma

Let $f_W(x)$ be a ReLU neural network. The following structural property of the gradient holds.

Lemma 6 (Lemma 2.1 of [51]) *Let $f_W(x) = W_L\sigma(W_{L-1}\dots\sigma(W_1x)) : \mathbb{R}^d \to \mathbb{R}$ be a ReLU neural network. Then, we can write:*

$$\forall x \in \mathbb{R}^d : \sum_{i,j} \frac{\partial f_W(x)}{\partial W_k^{i,j}} W_k^{i,j} = \left\langle W_k, \frac{\partial f_W(x)}{\partial W_k} \right\rangle = f_W(x). \tag{49}$$

B Critical points of SGD

Consider the case of $\lambda = 0$,

$$\min_W L(f(W)) = \min_W \sum_{i=1}^N \ell_i^2 \tag{50}$$

with $\ell_i = y_i - f(W; x_i)$.

We minimize $L(f(W))$ by running the following dynamical system (e.g., gradient flow)

$$\dot{W} = \nabla_W L(f(W)) = \sum_i^N \nabla_W f(W; x_i)(y_i - f(W; x_i)). \tag{51}$$

SGD can be formulated as follows. First define

Definition 6 *A random vector $v \in R^d$ drawn from a distribution $\mathcal{D}$ is a sampling vector if $\mathcal{E}_{\mathcal{D}}[v_i] = 1, \forall i$*

Then, the stochastic version of Equation (50) is

$$\min_W \mathcal{E}_{\mathcal{D}}[L(f(W))] = \min_W \mathcal{E}_{\mathcal{D}} \sum_i^n v_i \ell_i \tag{52}$$

Usually the distribution over $\mathcal{D}$ is assumed to be random v with independent components v_i, satisfying condition 6. The literature gives the impression that in expectation SGD is equal to GD, which is true *if* ℓ_i^2 are quadratic functions.

Finding the interpolating global minimizers of $L = \sum \ell_i^2$ is equivalent to finding the set of network weights W^* that solve the system of equations $\ell_i(W^*) = 0, \forall i = 1, \cdots, N$. Thus instead of finding all the critical points of the gradient of L, we would like to find the joint minimizers – that is the W – that minimize $\ell_i^2, \forall i = 1, \cdots, n$.

We define *critical points of SGD* the solutions of $\ell_i \nabla \ell_i = 0, \forall i$. For a discussion see [36].

A critical point of SGD with minibatch of size 1 is defined as $\dot{z} = g(x_n) = 0, \forall n = 1, \cdots, N$. This compares with a critical point of GD defined as $\dot{z} = \sum_{n=1}^N g(x_n) = 0$. GD of course is SGD with minibatch size N. What about SGD with intermediate minibatch sizes? The answer is[15]

Lemma 7 *If $\dot{z} = \sum_{n=1}^M g(x_n) = 0$ for enough random SGD draws of size $M < N$, then $\dot{z} = g(x_n) = 0, \forall n = 1, \cdots, N$.*

C Lemma on ρ dynamics

Lemma 8 *Assume $\rho \sum \overline{f_n} < 1$ and a normalized network, that is $\|V_k\| = 1, \quad k = 1, \cdots, L$. Then the loss can be written as $\mathcal{L} = 1 - \rho(\frac{1}{2}\dot{\rho} + \mu)$.*

Proof

Consider the loss $\mathcal{L} = 1 - 2\rho\mu + \rho^2 M + \lambda\rho^2$ and $\dot{\rho} = 2\mu - 2\rho M - 2\lambda\rho$, which gives $2\rho M = 2\mu - 2\lambda\rho - \dot{\rho}$. Thus we obtain the result

$$\mathcal{L} = 1 - \frac{1}{2}\rho\dot{\rho} - \rho\mu. \tag{53}$$

[15] A caution here is necessary: the number of random draws of size M from a data set of size N is enormous since it is equal to $\binom{N}{M}$ and thus, though all of them provide an over-constrained set of equations, only a very small subset of meaningful constraints may be available in practice.

D Margins for SGD

Consider first a global minimum with $\lambda = 0$ and $\mathcal{L} = 0$. Such a global minimum corresponds to interpolation of all data points and to $\dot{\rho} = 0$. Thus $\rho_0 f_n = y_n = \pm 1$ and the margins for all n are the same. We assume that this is the minimum ρ at convergence with $\lambda = 0$ and thus with no restrictions on rank. The associated ρ is ρ_0. This also implies that there is no other solution for $\lambda = 0$ with f_n that are *all equally larger* than $\frac{1}{\rho}$.

Let us now assume that $\lambda > 0$. Consider two extreme situations.

- The size of the minibatch B is $B = N$. There is effectively no rank constraint wrt the N data points of a minibatch. Assume the margins all equal to each other as $\bar{f}_n = \mu = \frac{1}{\rho_0}, \quad \forall n, \sigma = 0$ and thus $M = \mu^2 = \frac{1}{\rho_0^2}$. At equilibrium of SGD then $\dot{\rho} = 0 = 2N\mu - 2N\rho M - 2\lambda\rho$ which yields

$$\rho_\lambda = \frac{\rho_0}{1 + \rho_0^2 \frac{\lambda}{N}}. \tag{54}$$

- The size of the minibatch B is $B << N$. In this case

$$\rho_\lambda = \frac{\rho_0}{1 + \rho_0^2 \frac{\lambda}{B}}, \tag{55}$$

  but this only holds for the B points of the minibatch.

The variance induced by the size of the minibatch increases with $\frac{\lambda\rho_0}{B}$. To see this, estimate the difference between the margins associated with the different ρ_λ for the two extreme cases we considered with minibatch size of B and of N. We obtain $\Delta\frac{1}{\rho_\lambda} \approx \frac{\rho_0^2\lambda(\frac{1}{B} - \frac{1}{N})}{\rho_0} \approx \rho_0\lambda\frac{N-B}{NB}$. For Figure 7 this gives an estimate of about 310^{-3} for the standard deviation. This estimate is an order of magnitude larger than the measured standard deviation.

E Unnormalized vs Normalized dynamics

As shown in E.1, the equilibria with and without normalization are the same for ρ and V_k but the dynamics is different. Consider Figure 1. Assume that the network on Figure 1a is un-normalized, that is optimized via GD without Lagrange multipliers and the network in Figure 1b is normalized, that is optimized via GD with the Lagrange multiplier term. For 1a, consider, for simplicity, the case in which all the norms ρ_k of the weight matrices $1, \cdots, L-1$ are initialized with the same value. Then because of Lemma 9 all the $\rho_k, \quad \forall k = 1, \cdots, L-1$ of Figure 1a will change together and remain equal to each other. Let us call $\rho_k = \rho_1$. It is then possible to consider ρ for the network of Figure 1b as $\rho = \rho_1^L$ (because of the homogeneity of the ReLUs) and look at its dynamics. Consider the case of $\lambda = 0$.

The equations for the un-normalized case are then

$$\dot{\rho} = 2L\rho^{\frac{2L-2}{L}} \left[\sum_n \bar{f}_n - \sum_n \rho(\bar{f}_n)^2 \right] \tag{56}$$

and

$$\dot{V}_k = -2\rho^{\frac{L-2}{L}} \sum_n \left(1 - \rho\bar{f}_n\right) \cdot \left(\frac{\partial \bar{f}_n}{\partial V_k} - V_k \bar{f}_n \right). \tag{57}$$

The equations for the normalized case (Figure 1b), in which we have a single ρ parameter, are

$$\dot{\rho} = 2[\sum_n \bar{f}_n - \sum_n \rho(\bar{f}_n)^2] \tag{58}$$

and

$$\dot{V}_k = 2\rho \sum_n \left[(1 - \rho\bar{f}_n)(V_k \bar{f}_n - \frac{\partial \bar{f}_n}{\partial V_k}) \right]. \tag{59}$$

Recall that for $\lambda = 0$, ρ_0 corresponds to the inverse of the margin: thus $\frac{1}{\rho_0} = f_n$, since f_n is the same for all n. Thus $\dot{V}_k$ is proportional to ρ (ρ is the inverse of the margin) in the normalized case and to $\rho^{\frac{L-2}{L}}$ in the un-normalized case. The proportionality factor combines with the learning rate when Gradient Descent replaces gradient flow. Intuitively, the strategy to decrease the learning rate when the margin is large seems a good strategy, since large margin corresponds to "good" minima in terms of generalization (for classification).

E.1 Unnormalized dynamics

We consider the dynamical system induced by GD on a deep net with ReLUs (see Figure 1a). We change variables by using [16] $W_k = \rho_k V_k$, $\|V_k\| = 1$. Following the calculations in [6], the following identities hold: $\frac{\partial \rho_k}{\partial W_k} = V_k^T$ and $\frac{\partial g_n}{\partial W_k} = \frac{\rho}{\rho_k}\frac{\partial f_n}{\partial V_k}$.

Thus gradient descent on $L = \mathcal{L} = \sum_n (\rho f_n - y_n)^2$ with the definitions of V_k and ρ_k yields the dynamical system (with $\dot{W}_k = -\frac{\partial L}{\partial W_k}$)

$$\dot{\rho_k} = \frac{\partial \rho_k}{\partial W_k}\dot{W}_k = V_k^T \dot{W}_k = -2\sum_n (\rho_k^L f_n - y_n) f_n \rho_k^{L-1} = -2\rho_k^{L-1}[\sum_n \rho_k^L (f_n)^2 - \sum_n f_n y_n] \tag{60}$$

and, with $S_k = I - V_k V_k^T$,

$$\dot{V}_k = \frac{\partial V_k}{\partial W_k}\dot{W}_k = \frac{S_k}{\rho_k}\dot{W}_k = -2\frac{\rho}{\rho_k^2}\sum_n (\rho f_n - y_n)(\frac{\partial f_n}{\partial V_k} - V_k f_n). \tag{61}$$

E.2 Equal growth

If we assume that all the ρ_k are the same at initialization, we can use the following lemma to show that all ρ_k are the same at all times :

Lemma 9 [6] $\frac{\partial \rho_k^2}{\partial t}$ *is independent of* k *for* $\lambda = 0$ *and no normalization.*

Proof
Start from $\mathcal{L} = \frac{1}{N}\sum_n (1 - \rho \bar{f}_n)^2 + \lambda \sum_k \|W_k\|^2$ with $\lambda = 0$. Then

$$\dot{W}_k = -\frac{2}{N}\sum_n (1 - \rho \bar{f}_n)\rho \frac{\partial \bar{f}_n}{\partial W_k} \tag{62}$$

and, since $\|\dot{W_k}\|^2 = W_k \dot{W_k}$, we write

$$\|\dot{W_k}\|^2 = -\frac{2}{N}\sum_n (1 - \rho \bar{f}_n)\rho \bar{f}_n \tag{63}$$

which is independent of k.

Notice that if $\lambda > 0$, then for weight decay applied to all layers (that is with a regularization term of the form $\lambda \sum_k \rho_k^2$)

$$\|\dot{W_k}\|^2 = -\frac{2}{N}\sum_n (1 - \rho \bar{f}_n)\rho \bar{f}_n - \lambda W_k \tag{64}$$

Thus the equal growth lemma 9 does not hold in the case of weight decay. Furthermore it does not hold in the case that the regularization term is $\lambda\rho^2$ with $\rho = \Pi_k \rho_k$.

[16] Changing coordinates from W_k to V_k, we can convert the previous dynamical system to one in ρ, V_k. Using some basic vector calculus to get $\frac{\partial \rho_k}{\partial t} = \frac{1}{\rho_k}\left\langle W_k, \frac{\partial W_k}{\partial t}\right\rangle$ and $\frac{\partial V_k}{\partial t} = \frac{1}{\rho_k}\left(I - V_k V_k^\top\right)\frac{\partial W_k}{\partial t}$.

E.3 Equal weight norms at all layers

Assume the setup of the previous section. Then $\rho = \rho_k^L$, where L is the number of layers.

We use Equation (60) to derive the dynamics of $\rho = \rho_k^L$ in terms of $\dot{\rho} = \sum_k \frac{\partial \rho}{\partial \rho_k} \dot{\rho_k}$. Thus,

$$\dot{\rho} = 2L\rho^{\frac{2L-2}{L}} [\sum_n f_n y_n - \sum_n \rho (f_n)^2] \tag{65}$$

which is an equation of the type known as "differential logistic equation" used for instance to model sigmoidal population growth. It has an interesting dynamics as shown in the simplified simulations of Figure 16. The equilibrium value for $\dot{\rho_k} = 0$ is

$$\rho_0 = \frac{\sum_n \bar{f}_n}{\sum_n f_n^2}. \tag{66}$$

Similarly, for V_k:

$$\dot{V}_k = -2\rho^{\frac{L-2}{L}} \sum_n (\rho f_n - y_n)(\frac{\partial f_n}{\partial V_k} - V_k f_n). \tag{67}$$

At equilibrium for V_k – that is when $\dot{V}_k = 0$ – the equation gives (with $\ell_n = \rho f_n - y_n$ and assuming $\sum f_n \ell_n \neq 0$)

$$\sum_n (\rho f_n - y_n) \frac{\partial f_n}{\partial V_k} = \sum_n (\rho f_n - y_n)(V_k^0 f_n). \tag{68}$$

F BN, GD, LM: remarks

F.1 Remarks on BN

Without BN and without WD $\frac{\partial g_n(W)}{\partial W_k} = \frac{\rho}{\rho_k} \frac{\partial f_n(V)}{\partial V_k}$; with BN but without weight decay this becomes $\frac{\partial g_n(W)}{\partial W_k} = \rho \frac{\partial f_n(V)}{\partial V_k}, \quad \forall k < L$ and $\frac{\partial g_n(W)}{\partial W_L} = \frac{\partial f_n(V)}{\partial V_k}$.

This dynamics can also be written as $\dot{\rho_k} = V_k^T \dot{W}_k$ and $\dot{V}_k = \rho S \dot{W}_k$ with $S = I - V_k V_k^T$. This shows that if $W_k = \rho_k V_k$ then $\dot{V}_k = \frac{1}{\rho_k} \dot{W}_k$ as mentioned in [32].

F.2 Lagrange multiplier vs Batch Normalization

The constrained Lagrange dynamics will not change the initial norm of the $L-1$ layers: to ensure that $\|V_k\| = 1$ the initial value of the $L-1$ weight matrices must be $\|V_k\| = 1, \quad k = 1, \cdots, L-1$.

In our toy model the dynamics above with Lagrange multipliers captures what is commonly believed to be the key normalization property of batch normalization. This property can be summarized by the fact that the output of unit i given by $(Wh)_i$, where h is the vector of activities of the units in the previous layer, does not change after normalization if W is replaced by scaled version cW. It is important to emphasize, however, that the Lagrange dynamics does not reflect several aspects of Batch Normalization. In particular, in our model the weight matrices at each layer are normalized whereas in Batch Normalization each unit activity in each layer is normalized. Thus the dynamics of the weights in our model is different from the dynamics under Batch Normalization (see discussions in [6] and also [32]). The equations in the main text involving V_k can be read in this way, that is restricted to each row. The normalization of each weight matrix yields $\nu_k = -\sum_n (\rho^2 f_n^2 - \rho y_n f_n)$. Batch normalization normalizes $\sum_j W_{i,j}$, that is the input to each unit h_i before the ReLU ($h_i = \sigma(\sum_j W_{i,j})$) over each minibatch. Since this is equivalent to normalizing $\sum_j W_{i,j}, \quad \forall i$, the whole matrix W is normalized.

Notice also that in the model of Figure 1b, we regularize a single ρ at the top of the network, whereas in the standard usage of BN each layer norm $\rho_k, \quad \forall k = 1, \cdots, L$ is subject to weight decay. All layers $k = 1, \cdots, L-1$, but the last one, are also subject to BN.

In summary, the Lagrange multiplier normalization is very similar[6] to Weight Normalization but is different in several aspects wrt Batch Normalization. We believe however that both capture the key property of normalization techniques – the invariance with respect to scaling the weight matrices.

F.2.1 Experiments with Batch Normalization

The model architecture used for these experiments contains three convolutional layers (the number of channels are 32, 64, 128 and 128, the filter size is 3×3), and two fully connected layers at the end with hidden sizes 1024 and 2, respectively. Following each convolution layer, we applied a ReLU nonlinearity and Batch Normalization. Batch Normalization is used with learnable "affine" shifting and scaling parameters turned off (since they can always be learned by the next layer). The weight matrices of all layers are initialized with zero-mean normal distribution, scaled by a constant, such that the Frobenius norm of each matrix is one of the initialization value set {0.8, 0.9, 1, 1.2, 1.3, 1.5}. The network was trained using square loss and SGD with batch size 128, momentum 0.9, Weight Decay ($\lambda = 0.001$ or 0), cosine annealing learning rate scheduler with intial learning rate 0.001 for 1000 epochs and no data augmentation. Every input to the network is scaled such that it has norm ≤ 1. The plots in Figure 18 and Figure 17 are based on a single run. ρ is computed as the product of the Frobenius norms of each weight matrix (recall that BN normalizes the unit activities and not the weights as LM does).

G More about normalization (Figure 1a)

In our toy model, we considered the situation of Figure 1b, where only the top layer weight matrix is not normalized and has ρ at the top, subject to weight decay, while the previous layers are all normalized but are not subject to weight decay. This model neatly separates layers that are normalized from layers that have weight decay; in this model no layer has weight decay *and* normalization.

However, in normal practice of training a deep network, the weight matrices W_k in all layers up to layer $L-1$ are subject to weight decay and normalization via batch norm; only the last layer weights W_L are not normalized but still subject to weight decay. In this section, we consider this case, using Lagrange multipliers in place of batch norm.

The usual training of a deep net as in Figure 1a corresponds to minimizing the functional

$$\mathcal{L} = \sum_n (1 - \rho \bar{f}_n)^2 + \sum_{k=1}^{L-1} \nu_k \rho_k^2 (\|V_k\|^2 - 1) + \mu(\|V_L\|^2 - 1) + \lambda \sum_{k=1}^{L} \rho_k^2 \tag{69}$$

with $\rho_k^2 \|V_k\|^2 = 1$ for $k = 1, \cdots, L$ with the definition $\rho = \Pi_k \rho_k$. Notice that Equation (69) is different from Equation (1) for the toy model.

G.0.1 Gradient flow

Gradient flow in the model of Figure 1 is

$$\dot{\rho_k} = -\frac{\partial \mathcal{L}}{\partial \rho_k} = 2\frac{\rho}{\rho_k} \sum_n (1 - \rho \bar{f}_n) \bar{f}_n - 2\nu_k \rho_k \|V_k\|^2 - 2\lambda \rho_k \quad \forall k = 1, \cdots, L-1 \tag{70}$$

$$\dot{\rho_L} = -\frac{\partial \mathcal{L}}{\partial \rho_L} = 2\frac{\rho}{\rho_L} \sum_n (1 - \rho \bar{f}_n) \bar{f}_n - 2\lambda \rho_L \tag{71}$$

and for $k < L$

$$\dot{V_k} = -\frac{\partial L}{\partial V_k} = 2\rho \sum_n (1 - \rho \bar{f}_n) \frac{\partial \bar{f}_n}{\partial V_k} - 2\nu \rho_k^2 V_k \tag{72}$$

and

$$\dot{V_L} = -\frac{\partial L}{\partial V_L} = 2\rho \sum_n (1 - \rho \bar{f}_n) \frac{\partial \bar{f}_n}{\partial V_L} - 2\mu V_L \tag{73}$$

To find μ we multiply by V_L^T to get $0 = 2\rho \sum_n (1 - \rho \bar{f}_n) \bar{f}_n - 2\mu \|V_L\|^2$ which gives

$$\mu = \frac{\rho}{\|V_L\|^2} \sum_n (1 - \rho \bar{f}_n) \bar{f}_n. \tag{74}$$

To find ν_k we multiply by V_k^T and obtain $0 = \rho \sum_n (1 - \rho \bar{f}_n) \bar{f}_n - \nu_k \|V_k\|^2 \rho_k^2$ which gives

$$\nu_k = \frac{\rho}{\|V_k\|^2 \rho_k^2} \sum_n (1 - \rho \bar{f}_n) \bar{f}_n. \tag{75}$$

Thus the gradient flow is the following dynamical system for $k < L$

$$\dot{\rho_k} = -2\lambda \rho_k \quad \forall k = 1, \cdots, L-1 \tag{76}$$

$$\dot{V}_k = 2\rho \sum_n (1 - \rho \bar{f}_n)(\frac{\partial \bar{f}_n}{\partial V_k} - V_k \bar{f}_n) \quad \forall k = 1, \cdots, L-1 \tag{77}$$

$$\dot{\rho_L} = 2\frac{\rho}{\rho_L} \sum_n (1 - \rho \bar{f}_n) \bar{f}_n - 2\lambda \rho_L \tag{78}$$

$$\dot{V}_L = 2\rho \sum_n (1 - \rho \bar{f}_n)(\frac{\partial \bar{f}_n}{\partial V_L} - V_L f_n) \tag{79}$$

The solution of Equation (76) is $\rho(t) = 1 - (1 - \rho_{t=0}) e^{-2\lambda t}$ because $\rho^2 = 1$ for $t \to \infty$.
An equivalent alternative and simpler formulation is to start from

$$\mathcal{L} = \sum_n (1 - \overline{g}_n)^2 + \sum_{k=1}^{L-1} \nu_k \|W_k\|^2 + \lambda \sum_{k=1}^{L} \|W_k\|^2 \tag{80}$$

under the constraint $\|W_k\|^2 = 1$.

Gradient flow in this model is

$$\dot{W}_k = 2 \sum_n (1 - \overline{g}_n) \frac{\partial \overline{g}_n}{\partial W_k} - 2\nu_k W_k - 2\lambda W_k \quad \forall k = 1, \cdots, L-1 \tag{81}$$

and

$$\dot{W}_L = 2 \sum_n (1 - \overline{g}_n) \frac{\partial \overline{g}_n}{\partial W_L} - 2\lambda W_L \tag{82}$$

To find ν_k we multiply by W_k^T and obtain $0 = \sum_n (1 - \rho \overline{g}_n) \overline{g}_n - \nu_k - \lambda$ which gives

$$\nu_k = \sum_n (1 - \rho \overline{g}_n) \overline{g}_n - \lambda. \tag{83}$$

Thus the gradient flow in W_k is the following dynamical system for $k < L$

$$\dot{W}_k = 2 \sum_n (1 - \overline{g}_n)(\frac{\partial \overline{g}_n}{\partial W_k} - W_k \overline{g}_n) \tag{84}$$

and

$$\dot{W}_L = 2 \sum_n (1 - \overline{g}_n) \frac{\partial \overline{g}_n}{\partial W_L} - 2\lambda W_L \tag{85}$$

This dynamics corresponds to the dynamics in ρ and V_k of section 4.

G.0.2 Gradient Descent with typical normalization

Here we take into account the effect of discretization for Figure 1a as we did in a previous section with η terms added to the left-hand side of the previous set of equations obtaining

$$\frac{\eta}{2} \ddot{W}_k + \dot{W}_k = 2 \sum_n (1 - \overline{g}_n)(\frac{\partial \overline{g}_n}{\partial W_k} - W_k \overline{g}_n) \tag{86}$$

and

$$\frac{\eta}{2} \ddot{W}_L + \dot{W}_L = 2 \sum_n (1 - \overline{g}_n) \frac{\partial \overline{g}_n}{\partial W_L} - 2\lambda W_L \tag{87}$$

H Experiments on generalization

As discussed in the main text, while training gives perfect classification and almost zero square loss, the margin on the training set decreases and the test error also increases with the percentage of random labels. This also happens with Batch Normalization as shown in Figure 19. These results are fully consistent with the generalization bounds Equations 47 and 48.

However, the margin does not explain the behavior shown in Figure 20 where small differences in margin are actually anti-correlated with small differences in test error. Tighter bounds (see [34]) may explain these small effects. Alternatively, if there exist several almost-interpolating solutions with the same norm ρ_0, they may have similar norm and similar margin but different ranks of the weight matrices.

I Additional derivations of Neural Collapse

In section 4 of the main paper we showed that the phenomenon of Neural Collapse can be derived from the critical points of gradient flow under the square loss with Weight Decay. In this section we present derivations of Neural Collapse in two more situations. In section I.1 we show that the critical points of gradient flow under weight normalization also exhibits Neural Collapse. In section I.2 we show that NC occurs in the case of deep networks trained with the exponential loss as well.

I.1 Normalization case

In this subsection we stick to the setting of section 4 of the main paper and consider a multiclass classification problem with C classes with a balanced training dataset $\mathcal{S} = \{(x_n, y_n)\}$ that has N training examples per class. We train a ReLU deep network $f_W : \mathbb{R}^d \to \mathbb{R}^C$, $f_W(x) = W_L\sigma(W_{L-1}\dots W_2\sigma(W_1 x)\dots)$ with Gradient Descent on the square loss with Weight Decay on the parameters of the network. Under the normalized parameterization, we have $f_W(x) = \rho f_V(x)$, where $f_V(x) = V_L\sigma(V_{L-1}\dots V_2\sigma(V_1 x)\dots)$ is the normalized network. We use the following array notation to denote the output vectors and the one-hot target vectors respectively: $f_V(x) = [f_V^{(i)}(x)]$, $y_n = [y_n^{(i)}]$. We will also follow the notation of [4] and use $h(x)$ to denote the last layer features of the deep network. This means that $f_V^{(c)}(x) = \langle V_L^c, h(x)\rangle$. Similar to section 4 of the main paper, we assume that the solution obtained by Gradient Descent satisfies the Symmetric Quasi-interpolation condition which we recall below:

Assumption 2 (Symmetric Quasi-interpolation) *Consider a C-class classification problem with inputs in a feature space $\mathcal{X}$, a classifier $f : \mathcal{X} \to \mathbb{R}^C$ symmetrically quasi-interpolates a training dataset $S = \{(x_n, y_n)\}$ if for all training examples $x_{n(c)}$ in class c, $f^{(c)}(x_{n(c)}) = 1 - \epsilon$, and $f^{(c')}(x_{n(c)}) = \frac{\epsilon}{C-1}$.*

As we mentioned, this condition follows from the dynamics under the square loss for the case of binary classification and thus is likely to hold also for the multi-class case. This brings us to the main result of this section:

Theorem 7 *For a ReLU deep network trained on a balanced dataset using gradient flow on the square loss with Weight Normalization and Weight Decay, critical points of Gradient Flow that satisfy Assumption 2 also satisfy the NC1-4 conditions for Neural Collapse.*

Proof Our training objective is $\mathcal{L}(\rho, V) = \frac{1}{2}\sum_{n=1}^{NC} \|y_n - \rho f_V(x_n)\|^2 + \frac{\lambda}{2}\rho^2$. We can use Gradient Flow with Weight Normalization algorithm to train the network. We can relate this algorithm to Gradient Flow on the regular network parameterization f_W through the following equation: $\frac{\partial V_k}{\partial t} = \rho_k S_k \frac{\partial W_k}{\partial t} = -\rho_k S_k \frac{\partial \mathcal{L}(W)}{\partial W}$, where $S_k = I - V_k V_k^\top$ For the parameters of the last layer, this translates to: $\frac{\partial V_L}{\partial t} = -\rho\left(I - [V_L^i (V_L^j)^\top]\right)\frac{\partial \mathcal{L}(W)}{\partial W_L}$. This means:

$$\frac{\partial V_L}{\partial t} = \rho\left[\sum_n [(y_n^{(i)} - \rho f_V^{(i)}(x_n))h(x_n)] - \sum_n \langle y_n - \rho f_V(x_n), f_V(x_n)\rangle V_L\right] \tag{88}$$

Now let us analyze the critical points of the dynamics of the last layer, considering each classifier vector V_L^c of V_L separately:

$$\sum_n \langle y_n - \rho f_V(x_n), f_V(x_n)\rangle V_L^c = \sum_n (y_n^{(c)} - \rho f_V^{(c)}(x_n))h(x_n) \tag{89}$$

Let us consider solutions that achieve *symmetric quasi-interpolation*, with $\rho f_V^{(c)}(x_{n(c)}) = 1 - \epsilon$, and $\rho f_V^{(c)}(x_{n(c')}) = \frac{\epsilon}{C-1}$. It is fairly straightforward to see that since $f_V^{(c)}$ and V_L^c do not depend on n, neither does $h(x_n)$, which shows NC1. Under the conditions of NC1 we know that all feature vectors in a class collapse to the class mean, i.e., $h(x_{n(c)}) = \mu_c$. Let us denote the global feature mean by $\mu_G = \frac{1}{C}\sum_c \mu_c$. This means we have:

$$\begin{aligned}\frac{CN}{\rho}\left(\epsilon(1-\epsilon) - \frac{\epsilon^2}{C-1}\right) V_L^c &= \epsilon N \mu_c - \frac{\epsilon N}{C-1}\sum_{j\neq c}\mu_j \\ &= \frac{\epsilon NC}{C-1}(\mu_c - \mu_G) \\ \implies V_L^c &= \frac{\rho}{1-\frac{C}{C-1}\epsilon}\frac{1}{C-1}(\mu_c - \mu_G)\end{aligned} \tag{90}$$

This implies that the last layer parameters V_L are a scaled version of the centered class-wise feature matrix $M = [\ldots \mu_c - \mu_G \ldots]$. Thus at equilibrium, with quasi interpolation of the training labels, we obtain $\frac{V_L}{\|V_L\|_F} = \frac{M^\top}{\|M|_F}$. This is the condition for NC3.

From the gradient flow 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:

$$\begin{aligned}\|V_L^c\|_2^2 &= \frac{\sum_n (y_n^{(c)} - \rho f_V^{(c)}(x_n)) f_V^{(c)}(x_n)}{\sum_n \langle y_n - \rho f_V(x_n), f_V(x_n)\rangle} \\ &= \frac{\frac{N\epsilon}{\rho}(1-\epsilon) - \frac{N\epsilon^2}{\rho(C-1)}}{\frac{CN}{\rho}\left(\epsilon(1-\epsilon) - \frac{\epsilon^2}{C-1}\right)} \\ &= \frac{1}{C}\end{aligned} \tag{91}$$

From the quasi-interpolation of the correct class label we have that $\langle V_L^c, \mu_c\rangle = \frac{1-\epsilon}{\rho}$ which means $\langle V_L^c, \mu_G\rangle + \langle V_L^c, \mu_c - \mu_G\rangle = \frac{1-\epsilon}{\rho}$. Now using Equation (90)

$$\begin{aligned}\langle V_L^c, \mu_G\rangle &= \frac{1-\epsilon}{\rho} - \frac{\left(1-\frac{C}{C-1}\epsilon\right)(C-1)}{\rho}\|V_L^c\|_2^2 \\ &= \frac{1-\epsilon}{\rho} - \frac{\frac{C-1}{C} - \epsilon}{\rho} = \frac{1}{\rho C}.\end{aligned} \tag{92}$$

From the quasi-interpolation of the incorrect class labels, we have that $\langle V_L^c, \mu_{c'}\rangle = \frac{\epsilon}{\rho(C-1)}$, which means $\langle V_L^c, \mu_{c'} - \mu_G\rangle + \langle V_L^c, \mu_G\rangle = \frac{\epsilon}{\rho(C-1)}$. Plugging in the previous result and using (91) yields

$$\begin{aligned}\frac{(C-1)\left(1-\frac{C}{C-1}\epsilon\right)}{\rho} \times \langle V_L^c, V_L^{c'}\rangle &= \frac{\epsilon}{\rho(C-1)} - \frac{1}{\rho C} \\ \implies \langle \tilde{V}_L^c, \tilde{V}_L^{c'}\rangle &= \frac{1}{\|V_L^c\|_2^2} \times \frac{-1}{C(C-1)} = -\frac{1}{C-1}\end{aligned} \tag{93}$$

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 [4]. ■

I.2 Exponential case

Here we show that in the case of binary classification with exponential loss (a proxy for logistic loss), we can derive the Neural Collapse properties if we assume SGD critical points (that is equilibria achieved

with SGD with minibatch size 1). We leave an extension of this calculation to the multi-class case to future work.

For the exponential loss with normalization and weight decay, the gradient flow corresponding to GD are $\frac{\partial \rho_k}{\partial t} = \frac{1}{N}\sum_n e^{-\rho y_n f_n}\bar{f}_n - \lambda\rho$ and $\frac{\partial V_k}{\partial t} = \rho\frac{1}{N}\sum_n e^{-\rho y_n f_n} y_n S_k \frac{\partial f_n}{\partial V_k}$.

The equations at equilibrium for the ρ flow are

$$\frac{1}{N}\sum_n e^{-\rho y_n f_n} y_n f_n - \lambda\rho = 0. \tag{94}$$

For GD, a critical point implies $\lambda\rho = \frac{1}{N}\sum_n e^{-\rho y_n f_n} y_n f_n$.

The condition above does not by itself imply that all the margins $y_n f_n$ are the same (which is required for NC1). However, let us add the separability assumption and the assumption of SGD equilibria in the flow of V_k and ρ: equilibria for SGD with minibatch[17] size of 1 implies that each of the terms should be vanishing independently, i.e.,

$$e^{-\rho y_n f_n} y_n f_n = \lambda\rho, \quad \forall n = 1, \cdots, N \tag{95}$$

and thus $e^{-\rho y_1 f_1} y_1 f_1 = e^{-\rho y_n f_n} y_n f_n, \forall n = 1, \cdots, N$. These transcendental equations suggest that $f_1 = \cdots = f_n$ because of the constraints on $y_n f_n$, such that $1 \geq y_n f_n > 0$. Hence, the margins $y_i f_i$ are all equal. Then $V_L h_{i,c} = f^c$ is also independent of i implying that $h_{i,c}$ is independent of i at convergence.

With this result, the other NC properties follow in similar way as in the square loss case.

From the $\frac{\partial V_L}{\partial t} = 0$ equation, we immediately get that at the critical points $h(x_+) \sim V_L f(x_+)$ and $h(x_-) \sim V_L f(x_-)$, giving us $h(x_+) = -h(x_-)$. This also implies that $\mu_+ = -\mu_-$ and $\mu_G = 0$. It follows then that, defining $M = [\mu_+ - \mu_G, \mu_- - \mu_G]$, NC3 follows as $\frac{W_L}{\|W_L\|_F} \equiv V_L = \frac{M^\top}{\|M\|_F}$.

The fact that $\mu_+ = -\mu_-$ also immediately gives us the equinorm property, and we get that $\langle \mu_+, \mu_- \rangle = -\frac{1}{2-1} = -1$ and hence $\langle \mu_c, \mu_{c'} \rangle = \frac{C}{C-1}\delta_{c,c'} - \frac{1}{C-1}$. Thus NC2 follows. As in the case of square loss, by the results of [4] we also get NC4, completing the proof.

The proof for the exponential loss requires SGD, unlike the square loss case. We do not know whether this is is just a technicality. *We conjecture that is not. In this case the prediction would be the NC1 should be found under the square loss case with or without SGD, whereas NC1 under the exponential loss requires SGD. We further conjecture that small minibatch sizes should be better than large ones for the exponential loss case.*

J More details on Neural Collapse

In this section we recap the details of Neural Collapse as described in [4]. It is an empirical phenomenon that has been observed in the terminal phase of training deep networks, which we can associate with training beyond the point of separation. We listed the four conditions that are associated with Neural Collapse in section 4 of the main paper. Here we present them in a more formal manner.

We first define a deep network $f_W(x) = W_L h(x)$, where $h(x) \in \mathbb{R}^p$ denotes the last layer features of the deep 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 $\{(x_n, y_n)\}$ with N samples per class. We can compute the per-class mean of the last layer features as:

$$\mu_c = \frac{1}{N}\sum_{n\in N(c)} h(x_n) \tag{96}$$

The global mean of all features can be computed as:

$$\mu_G = \frac{1}{C}\sum_c \mu_c \implies \mu_G = \frac{1}{NC}\sum_{n=1}^{NC} h(x_n) \tag{97}$$

[17] We conjecture that the argument is valid also for minibatch sizes larger than 1 but smaller than N, see [9].

The second order statistics of the last layer features can be computed as:

$$\Sigma_W = \frac{1}{C}\sum_{c=1}^{C}\frac{1}{N}\sum_{n\in N(c)}(h(x_n)-\mu_c)(h(x_n)-\mu_c)^\top$$
$$\Sigma_B = \frac{1}{C}\sum_{c=1}^{C}(\mu_c-\mu_G)(\mu_c-\mu_G)^\top \tag{98}$$
$$\Sigma_T = \frac{1}{NC}\sum_{n=1}^{NC}(h(x_n)-\mu_G)(h(x_n)-\mu_G)^\top$$

Where Σ_W is 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) $\Sigma_W \to 0$, or within-class variability of last-layer training activations collapses to zero.

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 $\left\|\frac{W^\top}{\|W\|_F} - \frac{M}{\|M\|_F}\right\| \to 0$, 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 $\text{argmax}_c\langle W_L^c, h(x)\rangle \to \text{argmin}_c\|h(x)-\mu_c\|_2$.

K Time evolution of ρ_k

The norms ρ_k change very little when $\lambda = 0$. This is true for BN and LM. For $\lambda > 0$ in the presence of BN, the ρ_k decrease over time, unlike the case of LM.

L Three stages of model (b) training

The total ρ increases first and then decrease to some value. At the epoch with the highest total ρ the network starts to achieve FULL training accuracy. We can distinguish three stages during training. In Stage I, ρ does not grow (between epoch 0-20 , for example, the blue and brown curves, training accuracy is at chance during this period). In stage II ρ starts to increase until some peak value at epoch 100 (for blue and brown curves), when the training accuracy is perfect. In stage III ρ start decreasing at epochs $100-550$.

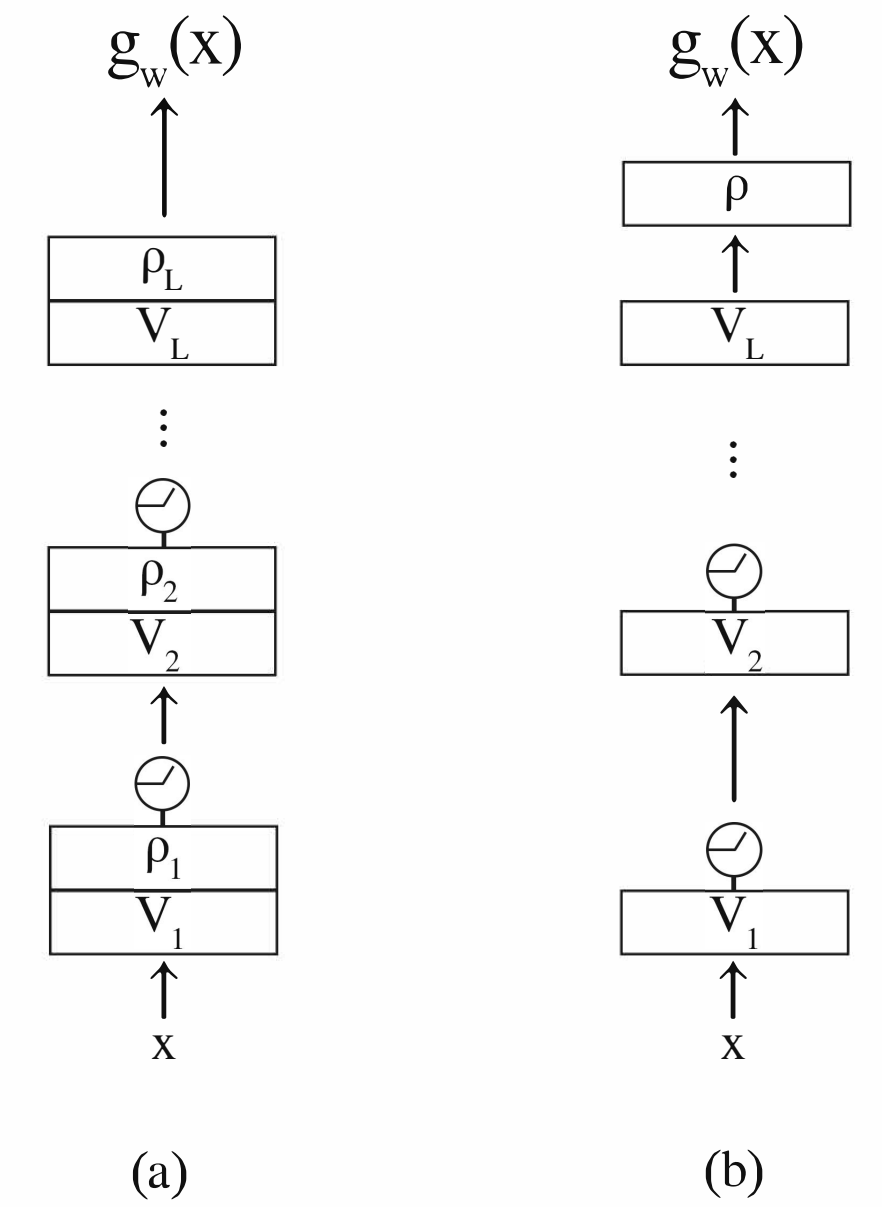


Figure 1: *Two parametrizations of a deep network. The circles represent the ReLU nonlinearity. Each box corresponds to a layer. We use network a) in the case in which the weight matrices $W_k = \rho_k V_k$ with $\|V_k\| = 1$ are not normalized by an algorithm like LM. We use network b) when the weight matrices V_k at each layer are actively normalized and only the last layer ($\rho_L V_L$) is not under normalization. Notice that Batch Normalization (see Figure 2) normalizes the activity of each neuron just before the ReLU nonlinearity. Normalizing the weight matrices, as weight normalization or LM do, is different, though both normalization techniques capture the relevant property of normalization – to make the dot product invariant to scale.*

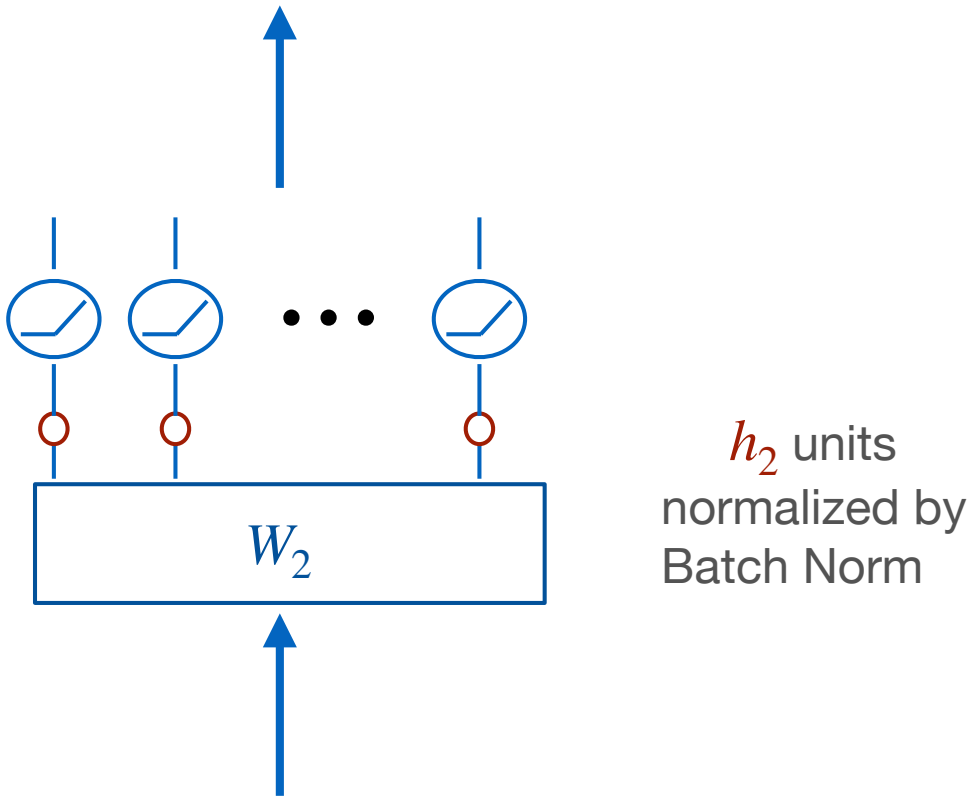


Figure 2: *Normalization by Batch norm acts on the units* (h_2) *in each layer before the ReLU nonlinearity on the activation of the units, but not directly on the weights* (W_2).

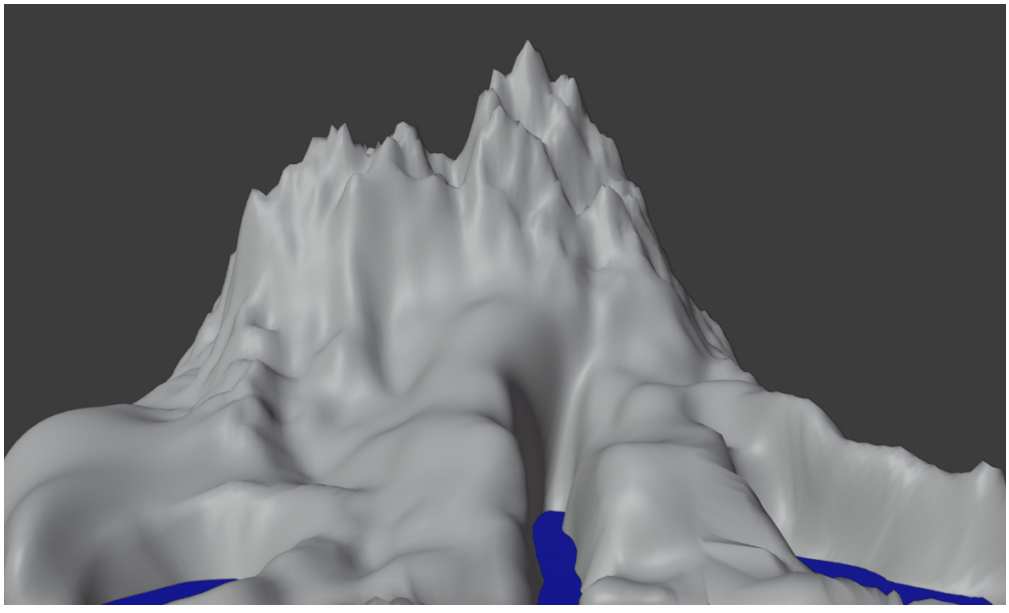


Figure 3: *A speculative view of the landscape of the loss with global degenerate valleys for* $\rho \geq \rho_0$ *with* V_1 *and* V_2 *weights of unit norm. Think of the loss as the mountain emerging from the water with zero-loss being the level of the water.* ρ *is the radial distance from the center of the mountain.*

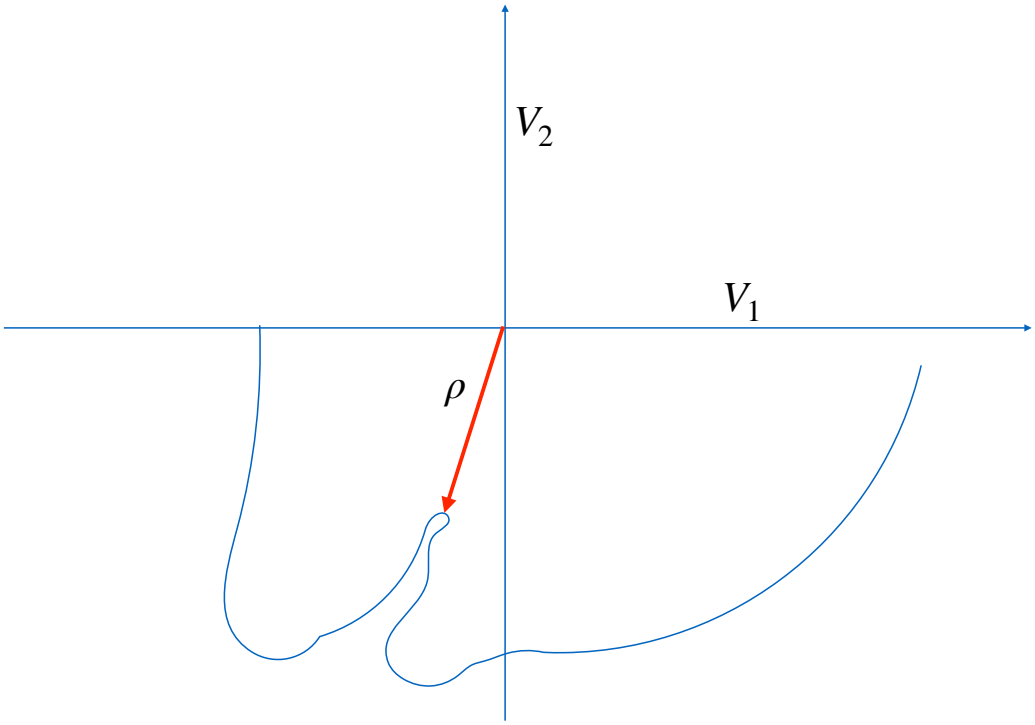
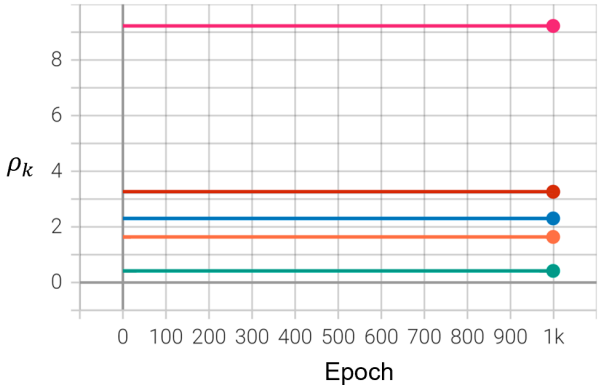


Figure 4: *The coastline of the loss (see previous figure) shows the boundaries of the zero loss degenerate minimum where $\mathcal{L} = 0$ in the high-dimensional space of ρ and $V_k \quad \forall k = 1, \cdots, L$. The degenerate global minimum is shown here as a connected valley outside the coastline. The red arrow marks the minimum with minimum ρ. Notice that, depending on the shape of the multidimensional valley, regularization may not guarantee convergence to the minimum norm solution, unlike in the linear network case.*



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 5: *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∼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.*

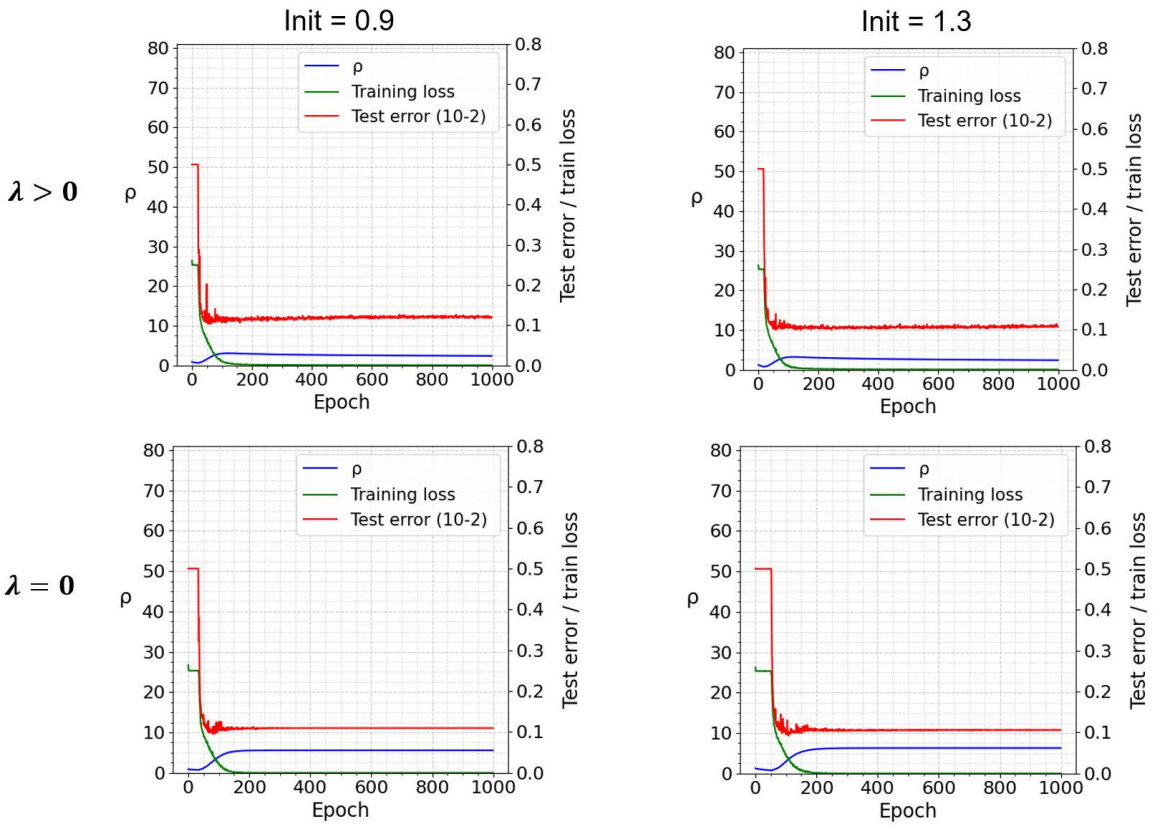


Figure 6: *Training dynamics of last layer norm ρ, training loss and test error over 1000 epochs with different initialization (0.9) in the first column and (1.3) in the second column. The first row is with Weight Decay $\lambda = 0.001$, and the second row is with Weight Decay $\lambda = 0$. The network was trained with Cosine Annealing learning rate scheduler (with initial learning rate $\eta = 0.03$, ending with $\eta = 0.0299$).*

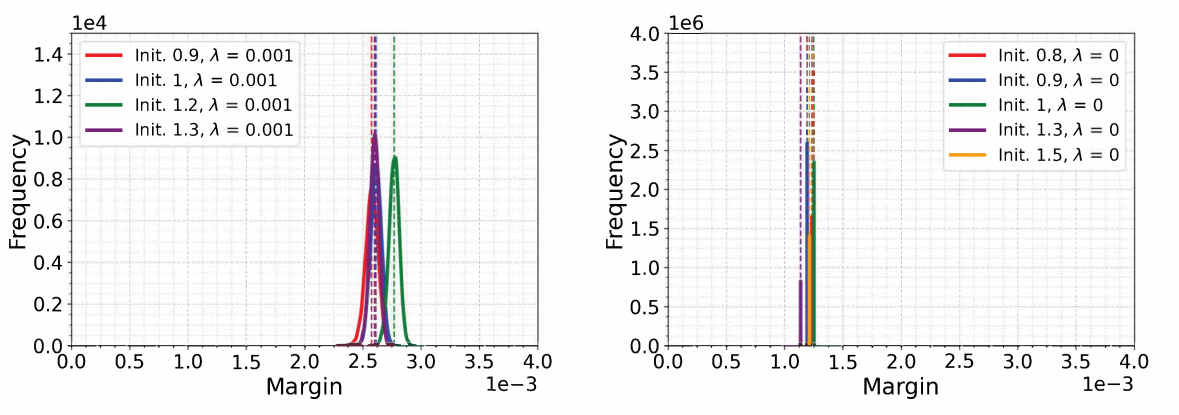


Figure 7: *Training margins computed over 10 runs for binary classification on CIFAR10 trained with square loss, Lagrange Multiplier normalization, and Weight Decay $(\lambda) = 0.001$ (left) and without Weight Decay (right, $\lambda = 0$) for different initializations ($init.$ = 0.8, 0.9, 1, 1.2, 1.3 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.*

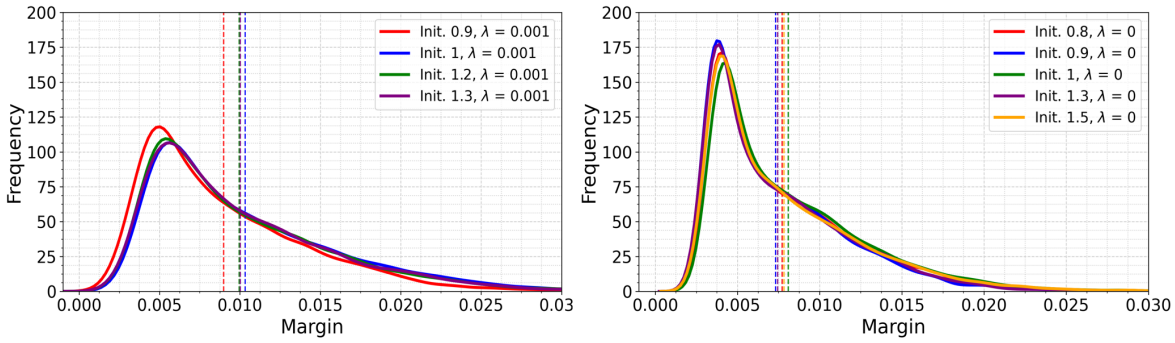


Figure 8: *Training margins for binary classification on the CIFAR10 dataset trained with cross-entropy loss, Lagrange Multiplier normalization and Weight Decay* $(\lambda) = 0.001$ *(left) and without Weight Decay (right,* $\lambda = 0$*) for different initializations* $(init. = 0.8, 0.9, 1, 1.2, 1.3$ *and* $1.5)$*. We applied a cosine learning rate scheduler with initial learning rate 0.01 during training. In the absence of weight decay* $(\lambda = 0)$ *there is clear evidence of SGD noise unlike in the square loss case.*

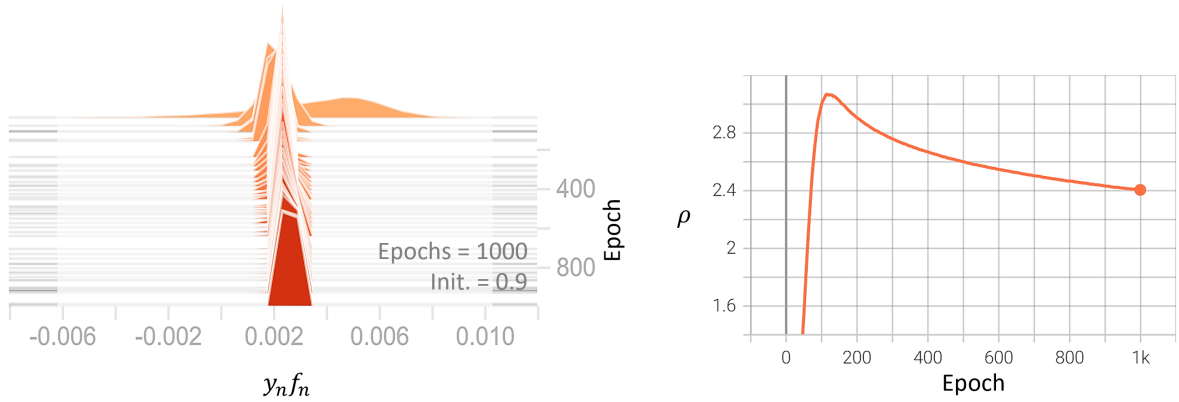


Figure 9: *Histogram of* $y_n f_n$ *across 1000 training epochs for binary classification on the CIFAR10 dataset with Lagrange Multiplier and weight decay* $(\lambda) = 0.001$*, initial learning rate 0.03, initialization 0.9. The histogram narrows as training progresses. The final histogram (in red) 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.*

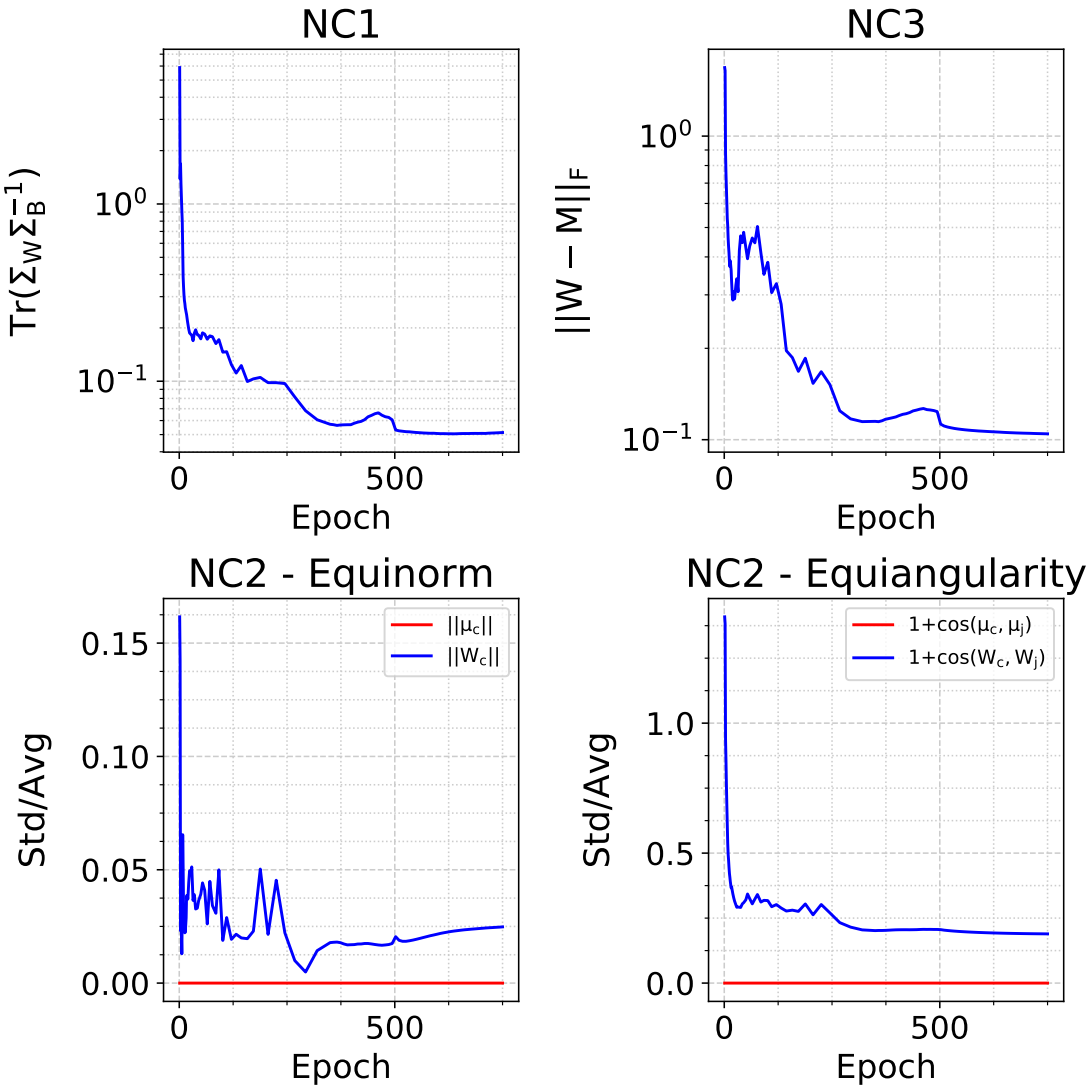


Figure 10: *Neural Collapse occurs during training for binary classification. The key conditions for Neural Collapse are: (i) NC1 - Variability collapse, which is measured by $Tr(\Sigma_W \Sigma_B^{-1})$, where Σ_W, Σ_B are the within and between class covariances, (ii) NC2 - equinorm and equiangularity 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.*

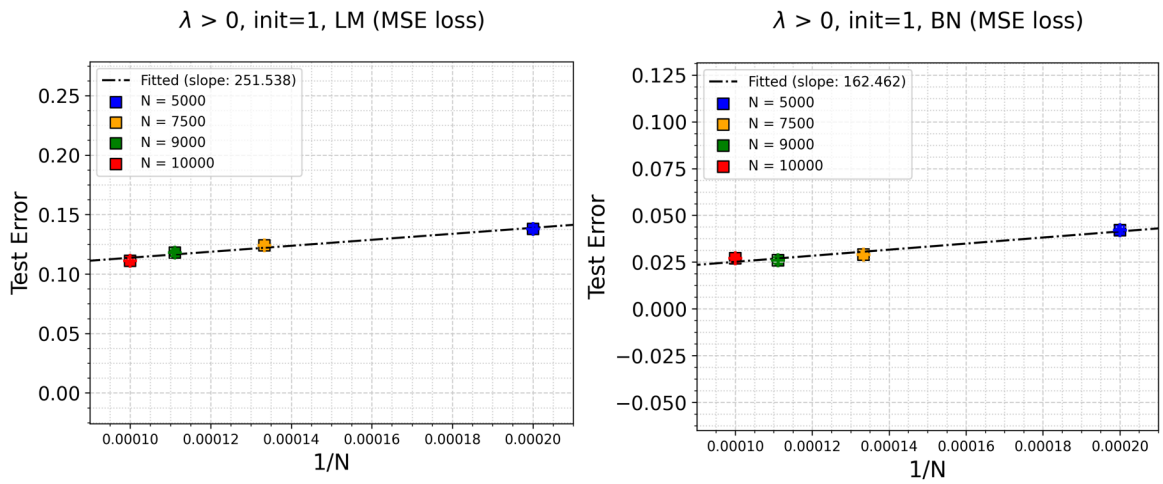


Figure 11: *Dependence on the number N of training examples of the test error. Binary classification on CIFAR10 trained with Lagrange Multiplier - LM (left) and with Batch Normalization - BN (right), the initialization scale is 1, both apply weight decay ($\lambda = 0.001$). The linear least square regression lines between the test error and $1/N$ are shown in the black dashed lines. (Left) the R-squared coefficient is 0.965 , p value is 0.018, slope: 251.54, y_intercept: 0.089. (Right) the R-squared coefficient is 0.957 , p value is 0.022, slope: 162.46, y_intercept: 0.009. The test errors with LM for $N = 5000, 7500, 9000, 10000$ are 0.138, 0.124, 0.118 and 0.111; the test errors with BN are 0.042, 0.029, 0.026 and 0.027. (NOT DONE!!!!!!)*

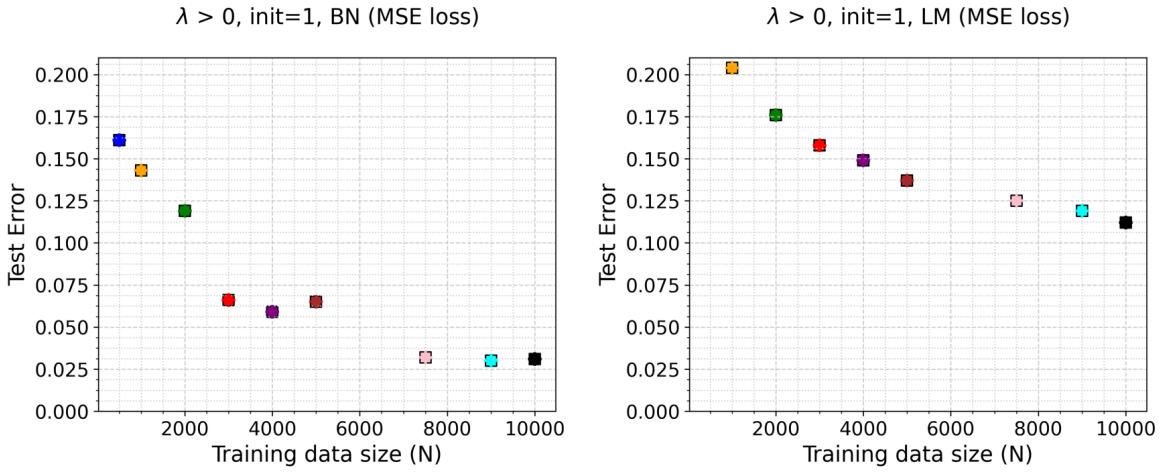


Figure 12: Test error versus total number of training data samples (N = 500, 1000, 2000, 3000, 4000, 5000, 7500, 9000 and 10000) with BN (left) and LM (right). For the experiments we applied the square loss, initialization 1 and weight decay ($\lambda > 0$), SGD optimizer with cosine annealing scheduler. All cases achieved 100% training accuracy but with different converge speed (the larger N, the faster the model converges). Specifically, the BN model achieved better test errors than the LM model.

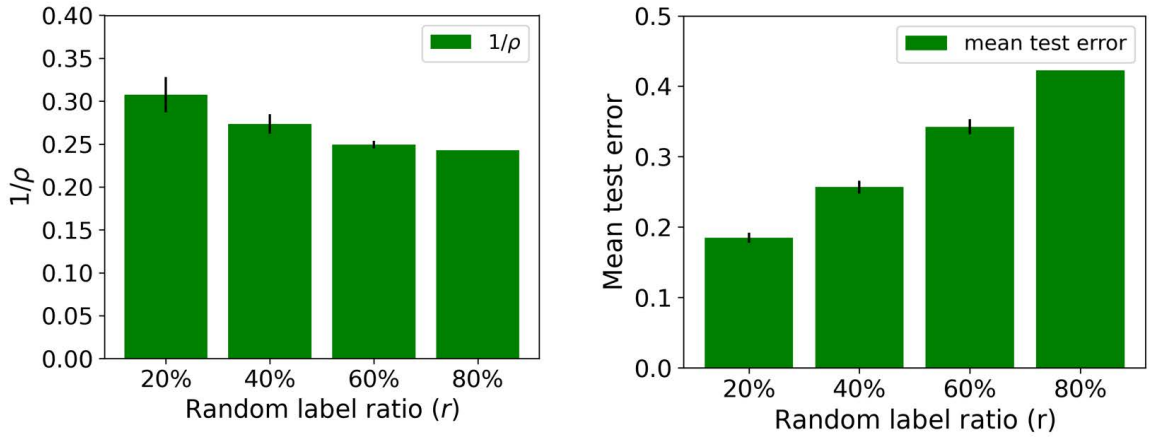


Figure 13: *Mean $1/\rho$ and test error results over 10 runs for binary classification on CIFAR10 trained with Lagrange Multiplier and different percentages of random labels ($r = 20\%, 40\%, 60\%$ and 80%), initialization scale 1 and weight decay 0.001. (Note that this network fails to get convergence with 100% random labels.)*

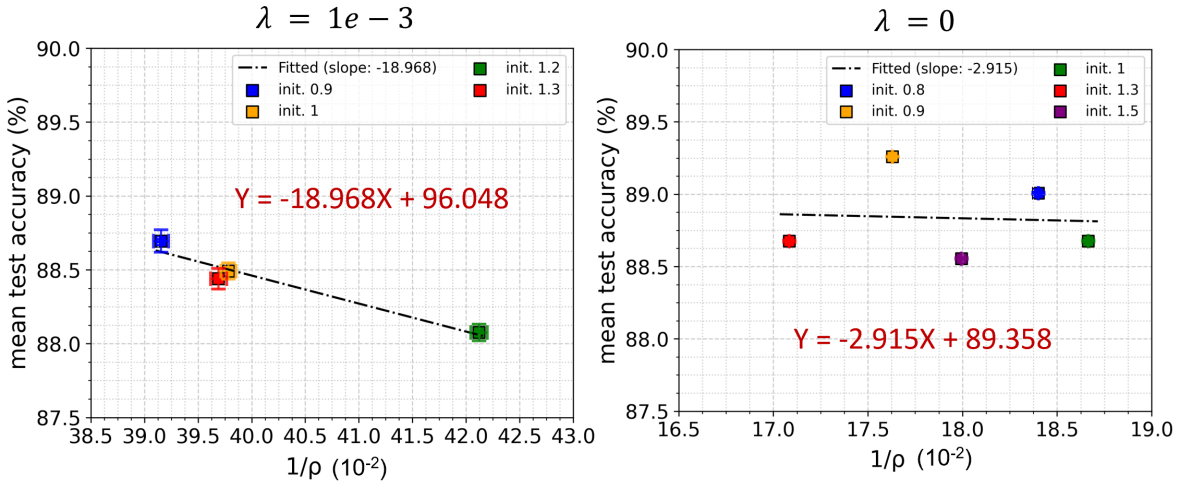


Figure 14: *Scatter plots for $1/\rho$ and mean test accuracy based on 10 runs for binary classification on CIFAR10 using Lagrange Multiplier (LM) normalization, square loss and Weight Decay (left) and without Weight Decay (right). In the left figure, the network was trained with different initialization 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.*

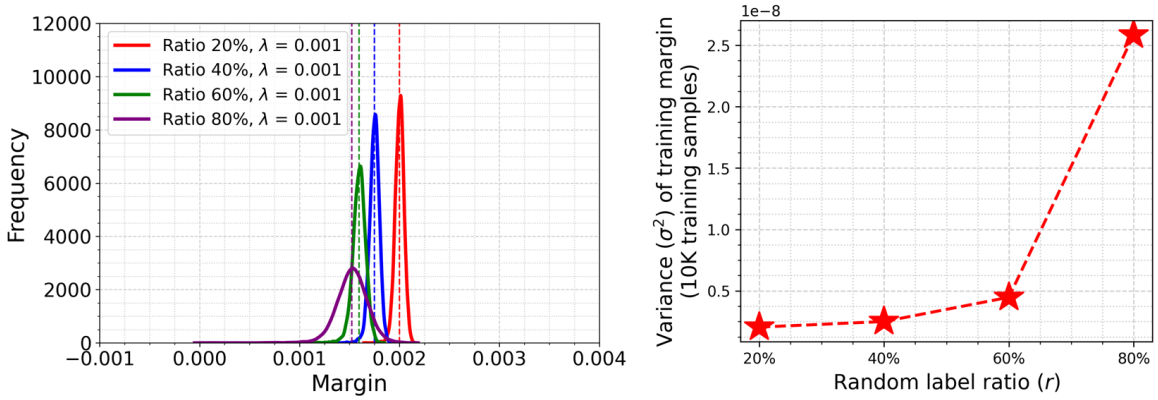


Figure 15: *Average training margin distribution (left) and the corresponding margin variance - σ^2 (right) over 10K training samples by 10 runs for binary classification on the CIFAR10 with four different random label ratios ($r = 20\%, 40\%, 60\%$ and 80%). The networks were trained with Lagrange Multiplier, weight decay ($\lambda = 1e-3$) and initialization scale 1. The higher the random label ratio is, the smaller the mean margins (vertical dashed lines in the left figure) and the greater the variance (right) of the training margins. Specifically, the mean margin (μ) decreases from 2.001e-3 (with 20% random labels) to 1.524e-3 (with 80% random labels); the variance (σ^2) grows from 2.04e-9 (with 20% random labels) to 2.58 e-8 (with 80% random labels).*

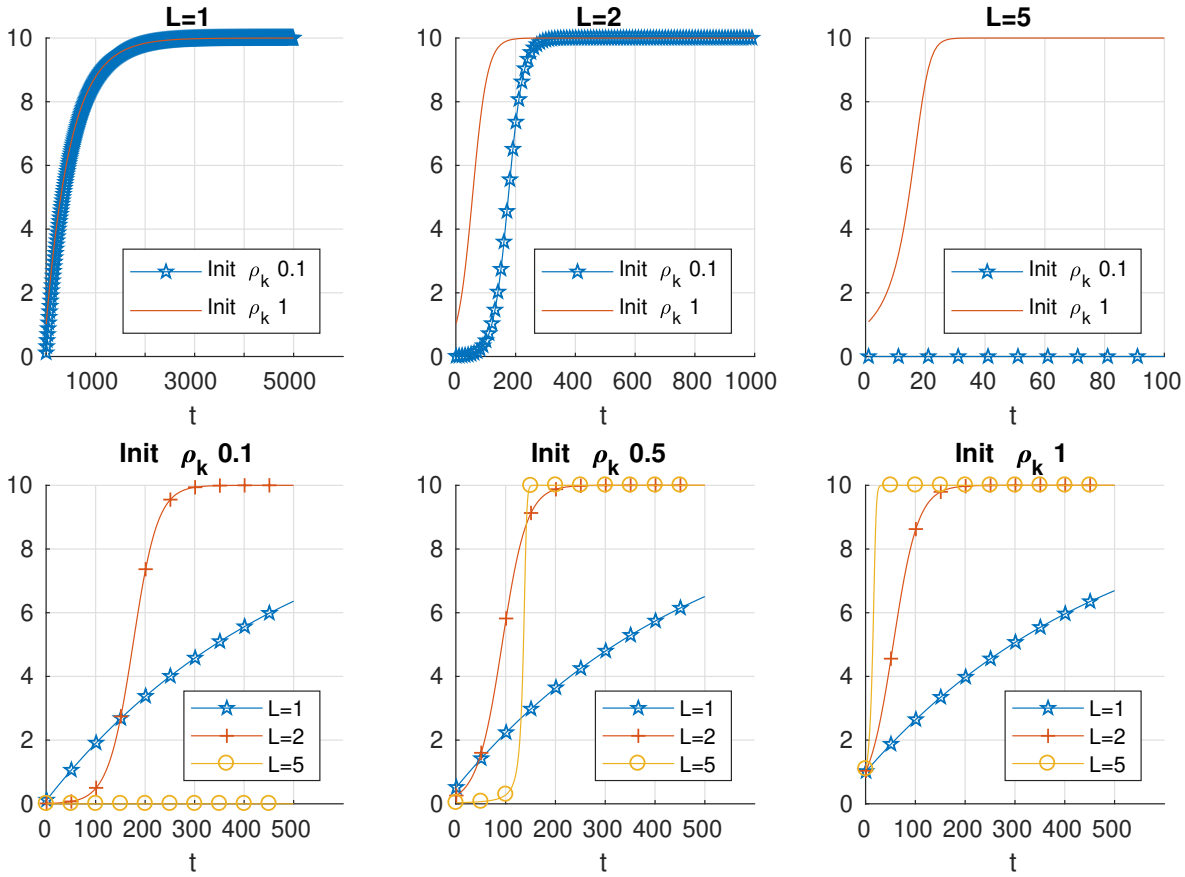


Figure 16: *Simulation of ρ from the logistic equation related to Equation (65), in which the terms $\sum y_n f_n$ and $\sum f_n^2$ are positive constants.*

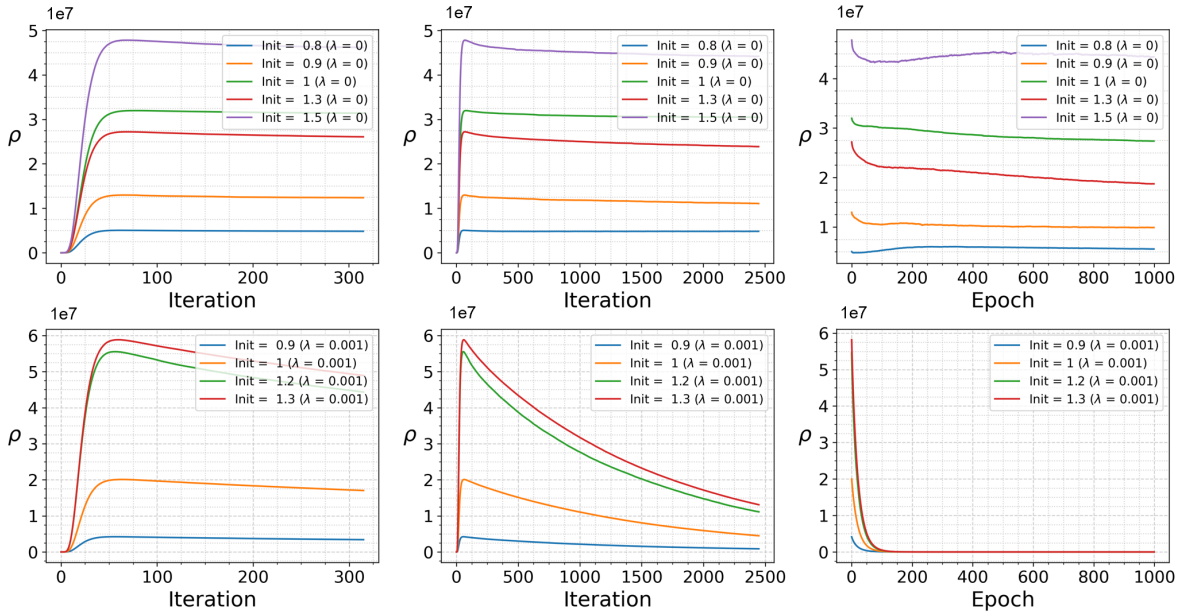


Figure 17: *Dynamics of ρ for binary classification on CIFAR10 trained with square loss, Batch Normalization and Weight Decay (Top) and without Weight decay (Bottom) for different initializations (0.8, 0.9, 1, 1.2, 1.3 and 1.5). The network is convolutional plus fully connected layers.*

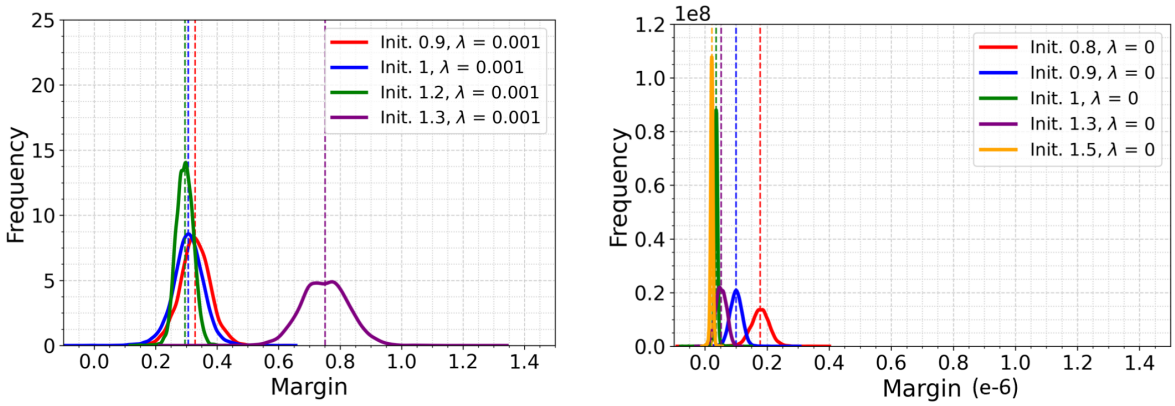


Figure 18: *Training margins for binary classification on the CIFAR10 dataset trained with square loss, Batch Normalization and Weight Decay (λ) = 0.001 (left) and without Weight Decay (right, $\lambda = 0$) for different initializations ($init.$ = 0.8, 0.9, 1, 1.2, 1.3 and 1.5). The network is convolutional plus fully connected layers. We applied a cosine learning rate scheduler with initial learning rate 0.01 during training. (Left) with weight decay the test errors for init. = 0.9, 1, 1.2, 1.3 are 0.024, 0.034, 0.026, 0.029. (Right) without Weight Decay the margins result in a range of smaller values, each with essentially zero variance; the corresponding test errors for $init.$ = 0.8, 0.9, 1, 1.3, 1.5 are 0.057, 0.057, 0.046, 0.057 and 0.055, respectively.*

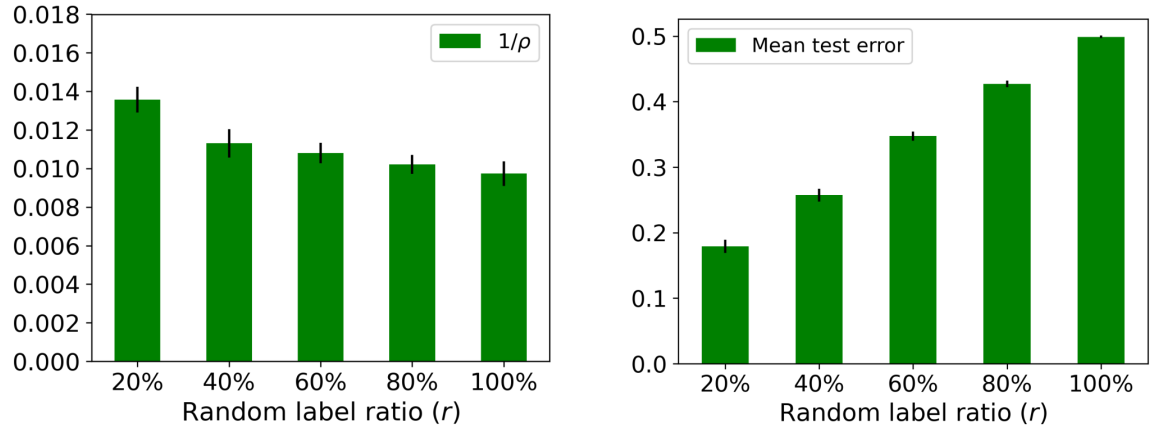


Figure 19: *Mean $1/\rho$ and test error results over 10 runs for binary classification on CIFAR10 trained with batch normalization and different percentages of random labels ($r = 20\%, 40\%, 60\%, 80\%$ and 100%), initialization scale 0.1 and weight decay 0.01.*

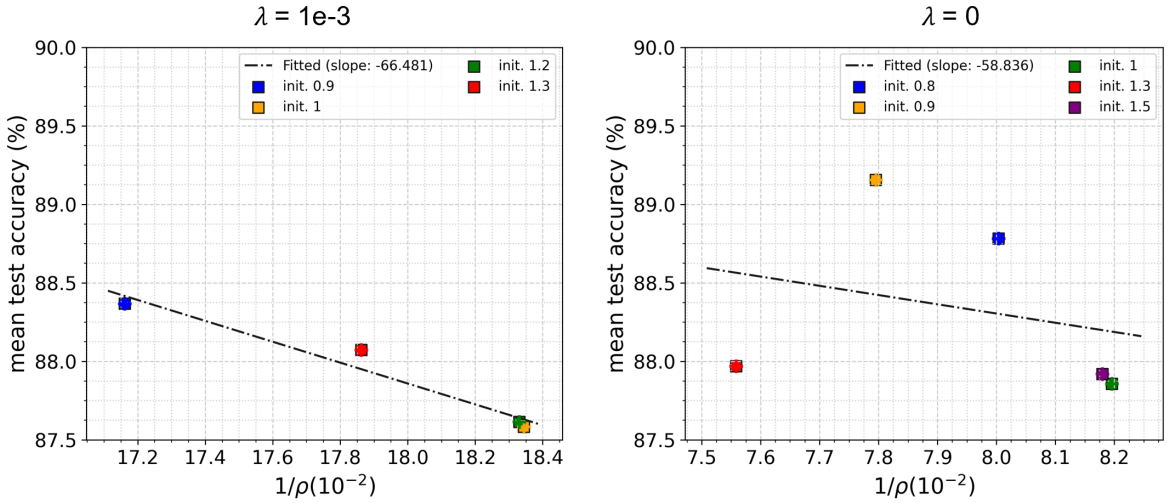


Figure 20: *Scatter plot for $1/\rho$ and mean test accuracy based on single run for binary classification on CIFAR10 using Lagrange Multiplier (LM) normalization, cross-entropy loss and Weight Decay (left) and without Weight Decay (right). In the left figure, the network was trained with different initialization scales (init. $= [0.9, 1, 1.2, 1.3]$) and with weight decay ($\lambda = 1e-3$); In the right figure, the network was trained with init. $= [0.8, 0.9, 1, 1.3, 1.5]$ and no weight decay ($\lambda = 0$). When $\lambda > 0$, the coefficient (R^2), p-value and slope for linear regression between $1/\rho$ and mean test accuracy are: $R^2 = 0.953$, p-value $= 0.024$, slope = -66.481; When $\lambda = 0$, the coefficient $R^2 = 0.072$, p-value $= 0.663$ and the slope = -58.836.*

init	mean test error (λ =0)	init	mean test error (λ>0)
0.8	0.057	0.9	0.024
0.9	0.057	1.0	0.034
1.0	0.046	1.2	0.026
1.3	0.057	1.3	0.029
1.5	0.055	NaN	NaN

Figure 21: The mean test error results for the model trained with BN, square loss with WD ($\lambda > 0$) and without WD ($\lambda = 0$) for binary classification of CIFAR10 dataset. The network achieved better test performance and smaller test errors with WD than without using WD.

init.	Total rho(λ = 0)	init.	Total rho (λ >0)
0.8	5552728.5	0.9	2.967
0.9	9891673.0	1.0	3.173
1.0	27369400.0	1.2	3.309
1.3	18731234.0	1.3	1.302
1.5	44428328.0	NaN	NaN

Figure 22: ρ for model (b) trained with **BN**, square loss with WD ($\lambda > 0$) and without WD ($\lambda = 0$) for binary classification of CIFAR10 dataset. ρ without the WD (2nd column) for different initializations (0.8, 0.9, 1, 1.3, 1.5) are much bigger than the one with WD (last column). The specific layer-wise ρ_k training dynamics for initialization (0.9) with WD and without WD are shown in the Fig. 24 and Fig. 25, respectively.

init.	mean total rho(λ = 0)	std(λ >0)	init.	mean total rho (λ >0)	std (λ >0)
0.8	818.991	94.987	0.9	383.908	13.491
0.9	851.707	120.345	1.0	379.397	22.497
1.0	806.348	83.889	1.2	358.348	18.203
1.3	884.604	70.555	1.3	382.984	36.802
1.5	828.836	63.238	NaN	NaN	NaN

Figure 23: ρ averaged over 10 runs for model (b) trained with **LM**, square loss with WD ($\lambda > 0$) and without WD ($\lambda = 0$) for binary classification of CIFAR10 dataset.

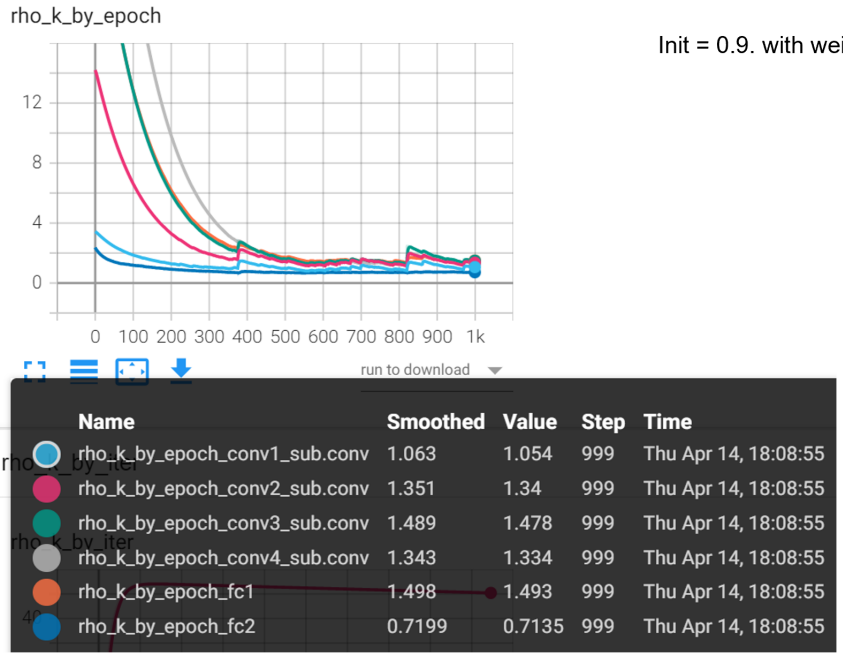


Figure 24: Layer-wise ρ_k dynamics during network training with BN, initialization 0.9, square loss and Weight Decay ($\lambda > 0$) for binary classification of CIFAR10 dataset. The weight norm of all layers decayed gradually by training epochs until to some small values.

Figure 25: Layer-wise ρ_k dynamics during network training with BN, initialization 0.9, square loss and no Weight Decay ($\lambda = 0$) for binary classification with CIFAR10 dataset. The weight norms (ρ_k) of all layers retain the initial values through the training epochs.

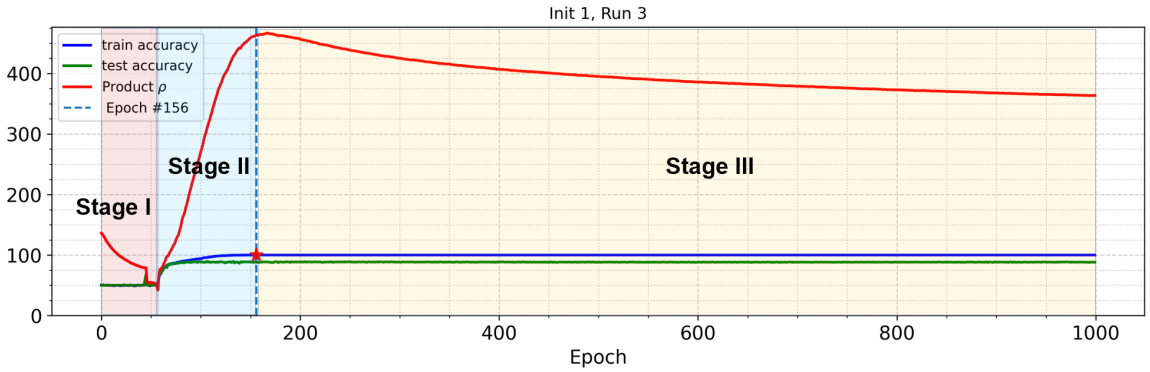


Figure 26: Product ρ, training accuracy and test accuracy during model training with LM, initialization 1, weight decay and square loss. It shows three obvious stages during LM model training. *Stage I*: the red $\rho = \prod_k \rho_k$ curve decreased a bit (between epoch 0-57) when the training accuracy and test accuracy are 50% during this period); *stage II*: Total ρ starts to increase to some peak value at epoch 156 when the training accuracy will increase from 50% to 100%; *Stage III*: ρ start to decrease to some value with fast speed in the very beginning from epoch 100 to epoch 550, then slow down after epoch 550. Moreover, there is no significant changes in both train and test errors at *stage III*.