



CENTER FOR  
**Brains  
Minds+  
Machines**

CBMM Memo No. 112

March 13, 2022

# Generalization in deep network classifiers trained with the square loss<sup>1</sup>

Tomaso Poggio, Qianli Liao and Mengjia Xu

## Abstract

Square loss has been observed to perform well in classification tasks. However, a theoretical justification is so far lacking, unlike the cross-entropy case. Here we discuss several observations on the dynamics of gradient flow under the square loss in ReLU networks. We show that convergence to a solution with the absolute minimum product  $\rho$  of the Frobenius norms of the weight matrices is expected when normalization techniques such as Weight Normalization or Batch Normalization (BN) are used together with Weight Decay (WD). In the absence of BN+WD, good solutions for classification may still be achieved because of the implicit bias towards small norm solutions in the GD dynamics introduced by close-to-zero initial conditions. A main property of the minimizers that bounds their expected error is the norm, since standard margin bounds show that for a fixed network architecture and the same training set, the interpolating solutions associated with a smaller value of  $\rho$  have better margin and thus better bounds on the expected classification error. The theory yields several predictions, including the role of normalization and weight decay. A main set of predictions is about Papanicolaou, Han and Donoho's Neural Collapse and the conditions under which Neural Collapse is expected to take place. Another interesting prediction is that regularization together with minimization of the loss under normalization of the weight matrices produces a bias towards rank one solutions for the weight matrices.



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

<sup>1</sup>This is the last version of a paper first published in August 2020; the main equations and the generalization approach were described in September 2020; the predictions about Neural Collapse for binary classification were first reported in November 2020; this version has corrected derivations of Neural Collapse for the multiclass, square loss case.

# Generalization in deep network classifiers trained with the square loss

Tomaso Poggio, Qianli Liao and Mengjia Xu

## Abstract

Square loss has been observed to perform well in classification tasks. However, a theoretical justification is lacking, unlike the cross-entropy [1] case for which an asymptotic analysis has been proposed (see [2] and [3] and references therein). Here we discuss several observations on the dynamics of gradient flow under the square loss in ReLU networks. We show that convergence to a solution with the absolute minimum norm is expected when normalization techniques such as Batch Normalization[4] (BN) or Weight Normalization[5] (WN) are used together with Weight Decay (WD) and small initialization of the weights. In the absence of BN+WD, good solutions for classification may still be achieved because of the implicit bias towards small norm solutions in the GD dynamics introduced by close-to-zero initial conditions. The main property of the minimizers that bounds their expected error is the norm, since standard bounds show that for a fixed network architecture, the interpolating solutions associated with a smaller value of the product of the Frobenius norms of the weight matrices have better margin and thus better bounds on the expected classification error. The theory yields several predictions, including the role of BN and weight decay. A main set of predictions is about Papanicolaou, Han and Donoho's Neural Collapse and the conditions under which Neural Collapse is expected to take place.

## 1 Introduction

### 1.1 Why square loss

We start from the assumption that an explanation of the ability of deep ReLU networks to be predictive, requires the identification of a mechanism of complexity control at work during the training of deep networks.

In the case of exponential-type loss functions such a mechanism has been identified in the asymptotic margin maximization effect of minimizing exponential-type loss functions [6, 2, 7]. However, this mechanism

- cannot explain the good empirical results that have been recently demonstrated using the square loss[8];
- cannot explain the empirical evidence that convergence for cross-entropy loss minimization depends on initialization.

This puzzle motivates our focus in this paper on the square loss.

Here we *assume* commonly used GD-based normalization algorithms such as BN (or WN) together with weight decay (WD), since such mechanisms seem essential for reliably training deep networks (and were used by [8])[9]. Crucially, our analysis depends on these assumptions<sup>1</sup>.

## 1.2 Regression and classification

In our analysis of the square loss, we need to explain when and why regression works well for classification, since the training minimizes square loss but we are interested in good performance in classification (for simplicity we consider here binary classification). A few preliminary remarks are helpful for understanding. Unlike the case of linear networks we expect several global zero square loss minima corresponding to interpolating solutions (in general degenerate, see [10] and reference therein). Although all interpolating solutions are optimal solutions of the regression problem, they will in general have different margins and thus different expected classification performance. In other words, zero square loss does not imply by itself neither large margin nor good expected classification. Why and when we expect the solutions of the regression problem to have large margin? We will show that the bias for large margin interpolating solutions depends on weight decay and initialization of the weights close to zero. As we will define later more formally, the function corresponding to a deep network can be written as  $g(x_n) = \rho f_n$  where  $g(x_n)$  is the output of the network for the training example  $x_n$ ,  $\rho$  is the product of the Frobenius norms of the weight matrices of the network and  $f_n = f(x_n)$  is the output of the normalized network for the input  $x_n$ . Notice that if  $g$  is a zero loss solution of the regression problem, then  $g(x_n) = y_n, \forall n$ . This is equivalent to  $\rho f_n = y_n$  where  $f_n$  is the *margin* for  $x_n$ . Thus the norm  $\rho$  of a minimizer is inversely related to its average margin (see Appendix C). In fact, for an exact zero loss solution of the regression problem, the margin is the same for all training data  $x_n$  and it is equal to  $\frac{1}{\rho_{eq}}$ . Notice that interpolating solutions are expected to be degenerate *for the regression problem* under overparametrization for each minimum: a possibly very small regularization is needed to guarantee convergence to the local minimum norm solution (which is maximum margin for classification) independently of initial conditions. As we will see in the next section, under the assumption of separability (with BN and weight decay or without), if  $\rho$  is small at initialization, it will grow monotonically under GD until a critical point of the gradient flow dynamics is reached. In other words, starting from small initialization, GD will explore critical points with  $\rho$  growing from zero. Thus quasi-interpolating solutions with small  $\rho_{eq}$  (corresponding to the best margin) may be found before large  $\rho_{eq}$  quasi-interpolating solutions which have worse margin (and are likely to be associated with the NTK regime). If the weight decay parameter is large enough, there may be independence from initial conditions. Otherwise, a small initialization is required, as in the case of linear networks, though the reason is quite different.

---

<sup>1</sup>We know that for overparametrized linear systems GD converges to the minimum norm solution if the weights are initialized close to zero values.

## 2 The dynamics of GD in $\rho$ and $V_k$

### 2.1 Notation

We define<sup>2</sup> a deep network with  $L$  layers with the usual coordinate-wise scalar activation functions  $\sigma(z) : \mathbf{R} \rightarrow \mathbf{R}$  as the set of functions  $g(W; x) = (W_L \sigma(W_{L-1} \cdots \sigma(W_1 x)))$ , where the input is  $x \in \mathbf{R}^d$ , the weights are given by the matrices  $W_k$ , one per layer, with matching dimensions. We sometime use the symbol  $W$  as a shorthand for the set of  $W_k$  matrices  $k = 1, \dots, L$ . There are no bias terms: the bias is instantiated in the input layer by one of the input dimensions being a constant. The activation nonlinearity is a ReLU, given by  $\sigma(x) = x_+ = \max(0, x)$ . Furthermore,

- we define  $g(x) = \rho f(x)$  with  $\rho$  defined as the product of the Frobenius norms of the weight matrices of the  $L$  layers of the network and  $f$  as the corresponding network with normalized weight matrices  $V_k$  (because the ReLU is homogeneous [7]);
- in the following we use the notation  $f_n$  meaning  $f(x_n)$ , that is the output of the normalized network for the input  $x_n$ ;
- we assume  $\|x\| = 1$  implying<sup>3</sup>  $|f(x)| \leq 1$  at convergence;
- the following structural property of the gradient of deep ReLU networks is useful (Lemma 2.1 of [11]):

$$\sum_{i,j} \frac{\partial g(W; x)}{\partial W_k^{i,j}} W_k^{i,j} = g(W; x); \quad (1)$$

for  $k = 1, \dots, L$ . Equation 1 can be rewritten as an inner product between  $W_k$  as vectors:

$$\left( W_k, \frac{\partial g(W; x)}{\partial W_k} \right) = g(W; x) \quad (2)$$

where  $W_k$  is here the vectorized representation of the weight matrices  $W_k$  for each of the different layers. We use this vectorized notation in a few places, hoping it will not confuse the reader. Notice that Equation 1 must be used with care: the  $W_k$  matrix depends on  $x$  and on the network!

- we assume that  $L \geq 2$ . The main reason is to avoid the case of linear networks with a unique minimizer of the square loss;
- *separability* is defined as correct classification for all training data, that is  $y_n f_n > 0, \quad \forall n$ . We call *average separability* when  $\sum y_n f_n > 0$ .

---

<sup>2</sup>For more details about basic properties, see [7].

<sup>3</sup>Because  $f(x)$  has the form of products of matrices of norm 1 (see Equation 63).

## 2.2 Gradient descent

The natural approach to training deep networks for binary classification using the square loss is to use stochastic gradient descent to find the weights  $W_k$  that minimize  $\mathcal{L} = \frac{1}{N} \sum_n \ell_n^2 = \frac{1}{N} \sum_n (g(x_n) - y_n)^2$ , with  $y = \pm 1$ . In this note, we consider the *gradient flow* associated with gradient descent.

## 2.3 Dynamics under normalization and weight decay

In the following we abuse the notation in order to keep it lighter in the following way. We denote  $f$  the network with weight matrices  $V_k$ . The weight matrices  $V_k$ ,  $k = 1, \dots, L - 1$  are not normalized but are constrained during gradient descent to converge to unit norm matrices; the last weight matrix  $W_L$  in the network  $f$  is not normalized. We define it as  $W_L = \rho V_L$ . Thus  $f$  here, differently from the previous section, is not the normalized network but a network that is supposed to converge to a network with  $L - 1$  normalized layers. We consider gradient descent on a modified loss

$$\mathcal{L} = \sum_n (\rho f_n - y_n)^2 + \nu \sum_{k=1}^{L-1} \|V_k\|^2 \quad (3)$$

with the constraint  $\|V_k\|^2 = 1$ . In fact, the constrained 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  $\|V_k\| = 1$ ,  $k = 1, \dots, L - 1$ .<sup>4</sup>

The *key assumption* in this paper is that the dynamics above with Lagrange multipliers, captures the key normalization property of batch normalization, though not all of its details (see Appendix A and discussions in [7] and also [9]). Thus *we assume that for network trained with BN*, following the spirit of the analysis of [9],  $\rho_k = 1$ ,  $\forall k < L$  and  $\rho_L = \rho$  where  $L$  is the number of layers. It is important to observe here that batch normalization – unlike Weight Normalization – leads not only to normalization of the  $L - 1$  weight matrices but also to normalization of each row of the matrices [7] because it normalizes separately the activity of each unit  $i$  and thus – indirectly – the  $W_{i,j}$  for each  $i$  separately. This implies that each row  $i$  in  $(V_k)_{i,j}$  is normalized independently and thus the whole matrix  $V_k$  is normalized (assuming the normalization of each row is the same 1 for all rows). 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, as shown in Appendix 2.3,  $\nu = -\sum_n (\rho^2 f_n^2 - \rho y_n f_n)$ .

As we will show, the dynamical system associated with the gradient flow of the Lagrangian of Equation 3 is “singular”, in the sense that normalization is not guaranteed at the critical points. Regularization is needed, and in fact it is common to use in gradient descent not only batch normalization but also weight decay. Weight decay (see Appendix E.1) consists of a regularization term  $\lambda \|W_k\|^2$  added to the Lagrangian yielding

<sup>4</sup>This dynamics is equivalent to “Weight Normalization”, as proved in [7], for deep networks. This dynamics can 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 [9].

$$\mathcal{L} = \sum_n (\rho f_n - y_n)^2 + \nu \sum_k \|V_k\|^2 + \lambda \rho^2. \quad (4)$$

The associate gradient flow is then the following well-defined dynamical system

$$\dot{\rho} = -2[\sum_n \rho (f_n)^2 - \sum_n f_n y_n] - 2\lambda \rho \quad (5)$$

$$\dot{V}_k = 2\rho \sum_n [(\rho f_n - y_n)(V_k f_n - \frac{\partial f_n}{\partial V_k})] \quad (6)$$

where the critical points  $\dot{\rho} = 0$ ,  $\dot{V}_k = 0$  are not singular for any arbitrarily small  $\lambda > 0$ . For  $\lambda = 0$  the zero loss critical point is pathological, since  $\dot{V}_k = 0$  even when  $(V_k f_n - \frac{\partial f_n}{\partial V_k}) \neq 0$  implying that a un-normalized interpolating solution can satisfy the equilibrium equations. This is expected from the constrained dynamics which by itself constrains the norm of the  $V_k$  to not change during the iterations. 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  $\rho_{eq}$ , unlike the  $\lambda = 0$  case. As a side remark, SGD, as opposed to gradient flow, may help (especially with label noise) to counter the singularity of the  $\lambda = 0$  case, even without weight decay, because of the associated random fluctuations around the pathological critical point.

Observe that for  $\lambda = 0$ ,  $\dot{\rho} = 0$  for zero loss minima, that is for interpolating solutions. The converse is not true: there are critical points that correspond to local minima or saddle points. In the following, we focus on interpolating or quasi-interpolating solutions (when  $\lambda > 0$ ).

### 2.3.1 Equilibrium values

The equilibrium value at  $\dot{\rho}_k = 0$  is<sup>5</sup> (see Appendix E.1)

$$\rho_{eq} = \frac{\sum_n y_n f_n}{\lambda + \sum_n f_n^2}. \quad (7)$$

Observe that  $\dot{\rho} > 0$  if  $\rho$  is smaller than  $\rho_{eq}$  and if average separability holds. Recall also that zero loss “global” minima (in fact arbitrarily close to zero for small but positive  $\lambda$ ) are expected to exist and be degenerate [10] – in addition to other critical points.

If we assume that the loss (with the constraint  $\|V_k\| = 1$ ) is a continuous function of the  $V_k$ , then there will be at least one minimum of  $\mathcal{L}$  at any fixed  $\rho$ , because the domain  $V_k$  is compact. This means that for each  $\rho$  there is at least a critical point<sup>6</sup> of the gradient flow of  $V_k$ , implying that for each critical  $\rho$  for which  $\dot{\rho} = 0$ , there is at least one critical point of the dynamical system in  $\rho$  and  $V_k$ .

<sup>5</sup> Notice that  $\dot{\rho} = 0$  is equivalent to  $\sum \ell_n f_n = 0$ . Thus the two conditions together –  $\dot{\rho} = 0$  and  $\dot{V}_k = 0$  – imply  $\sum \ell_n \frac{\partial f_n}{\partial V_k} = 0$ .

<sup>6</sup> If  $V_k f_n = \frac{\partial f_n}{\partial V_k}$  then  $\dot{V}_k = 0$  but this is not a critical point for the system  $\rho, V_k$  unless  $\dot{\rho} = 0$ .

Around  $\dot{V}_k = 0$  we have

$$\sum_n (\rho f_n - y_n) \frac{\partial f_n}{\partial V_k} = \sum_n (\rho f_n - y_n) (V_k^{eq} f_n), \quad (8)$$

where the terms  $(\rho f_n - y_n)$  will be in general different from zero if  $\lambda > 0$ .

In appendix D we describe the interesting dynamics associated with the unnormalized case (when  $\rho_k$  are all the same  $\forall k, k = 1, \dots, L$ ). If during GD (with BN), other layers, in addition to the last, have  $\rho_k$  different from 1 (which happens in practice), the dynamics will show some of the interesting properties described in appendix D, which favor small initializations to reach solutions with greater margin. The dynamics of Equation 28 is that the smaller  $\rho_{t=0}$  is, the longer it takes to  $\rho$  to grow (this phenomenon becomes stronger with a larger number of layers  $L$ ). Thus  $\rho$  is constrained by the nonlinear dynamics to be very small for a transient phase  $T$  of GD iterations ( $T$  is longer with more layers and longer with smaller initialization).

Appendix E describes a few additional properties of the dynamics of the normalized weights. The conclusions of this analysis can be summarized in

**Observation 1** *Assuming average separability, and gradient flow starting from  $\rho = \epsilon$  with small  $\epsilon > 0$ ,  $\rho$  grows monotonically until a minimum is reached at which  $\rho_{eq} = \frac{\sum_n y_n f_n}{\lambda + \sum_n f_n^2}$ . In the limit  $\lambda = 0$ , the minimum of the square loss approaches zero, corresponding to exact interpolation of all the training data (we assume that local minima are avoided by using SGD).*

and

**Observation 2** *Minimizers with small  $\rho_{eq}$  correspond to large average margin  $\sum y_n f_n$ . In particular, suppose that the gradient flow converges to a  $\rho_{eq}$  and  $V_k^{eq}$  which correspond to zero square loss. Among such minimizers the one with the smallest  $\rho_{eq}$  (typically found first, even for  $\lambda = 0$ , during the GD dynamics when  $\rho$  increases from  $\rho = 0$ ), corresponds to the (absolute) minimum norm – and maximum margin – solutions.*

In general, the critical points of the  $V_k$  for the same  $\rho_{eq}$  corresponding to minima of the loss are typically degenerate (see references in [12]) with dimensionality  $W - N$ , where  $W$  is the number of weights in the network<sup>7</sup>. All of them will correspond to the same norm and all will have the same margin for all of the training points. The dynamic leading to  $\rho_{eq}$  and  $V_k^{eq}$  requires an analysis not just of gradient flow but of SGD and of the associated Fokker-Planck equation.

### 3 Dynamics

Consider a minimum of GD for which the square loss is close to zero and  $\dot{V}_k = 0$ . Clearly critical points of  $\rho$  cannot exist if  $\rho$  is too small<sup>8</sup>. Since usually the maximum output of a multilayer

<sup>7</sup>In addition there are local minima and other critical points.

<sup>8</sup> $\rho \geq 1$  for a critical point to exist because the critical point with smallest possible  $\rho$  is for  $\rho = 1, f_n y_n = 1$ .

network is  $\ll 1$ , the first critical point for increasing  $\rho$  will be when  $\rho$  becomes large enough to allow the following equation to have solutions

$$\sum_n y_n f_n = \rho(\lambda + \sum_n f_n^2). \quad (9)$$

If gradient flow starts from very small  $\rho$  and there is *average separability*,  $\rho$  increases monotonically until such a minimum is found<sup>9</sup>. This analysis is for gradient flow. A satisfactory theory requires an analysis of gradient descent along the lines of [9]. If  $\rho$  is large, then  $\dot{\rho} < 0$  and  $\rho$  will decrease until a minimum is found.

For large  $\rho$  and very small or zero  $\lambda$ , we expect several solutions under GD<sup>10</sup>. The existence of several solutions is related to arguments showing the existence of NTK-based solutions: intuitively the last layer is enough in an extreme case – if the last layer before the linear classifier is overparametrized wrt training data – to provide solutions for *any* set of random weights in the previous layers (for large  $\rho$  and small  $f_i$ ). Furthermore the intermediate layer 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  $\rho < \rho_{min}$  (which, in practice, means  $\rho_{min} \gg 1$ ). With increasing  $\rho$  from  $\rho = 0$  there will be zero square-loss degenerate (see [10]) minima with the minimizer representing an interpolating (for  $\lambda = 0$ ) or almost interpolating solution (for  $\lambda > 0$ )<sup>11</sup>. We expect, however, that for  $\lambda > 0$  there will be a bias towards the minimum  $\rho_{eq}$  within the local degenerate minimum. Under certain conditions all the global minima – associated to interpolating solutions – will be connected within a unique, large valley parametrized. The argument is based on Theorem 5.1 of [13]: *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, this implies that *zero-square-loss minima with different  $\rho$  are connected.* A connected single valley of zero loss *does not* however guarantee that *SGD with WD will converge to the minimum norm global minimum 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 correspond to monotonically increasing or decreasing  $\rho$ .

All these observations are also supported by our numerical experiments. Figure 1, 2, 3, 4 and 5 show the case of gradient descent with batch normalization and weight decay, which corresponds to a well-posed dynamical system for gradient flow; the other figures show the same networks and data with BN without WD and without both BN and WD. As predicted by the analysis, the case of BN+WD is the most well-behaved, whereas the others strongly depend on initial conditions.

<sup>9</sup>We disregard local minima because they do not appear to be a problem for SGD and because it is always possible to check the training loss and restart GD if necessary.

<sup>10</sup>It is interesting to recall [10] that for SGD – unlike GD – the algorithm will stop only when  $\ell_n = 0 \quad \forall n$ , which is the global minimum and corresponds to perfect interpolation. For the other critical points for which GD will stop, SGD will never stop but may fluctuate around the critical point.

<sup>11</sup>Notice that the equilibrium value of  $\rho$  is a measure of “sparsity”: small  $\rho$  corresponds to  $f_i$  being close to either 1 or zero ( $\rho$  is in the order of  $\frac{1}{f_i}$ ).



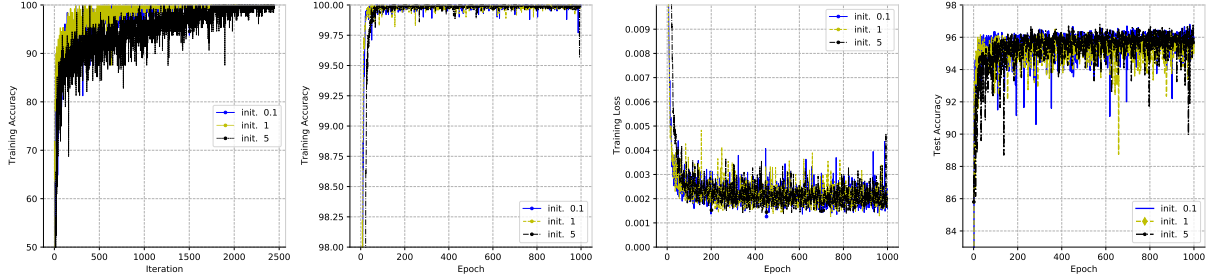


Figure 1: *ConvNet with Batch Normalization and Weight Decay Binary classification on two classes from CIFAR-10, trained with MSE loss. The model is a very simple network with 3 layers of convolutional Layers with number of channels being 32, 64 and 128. The last layer is a fully-connected classifier layer. ReLU nonlinearity is used. 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 5. We use weight decay of 0.01. We run SGD with batch size 128, constant learning rate 0.1 and momentum 0.9 for 1000 epochs. No data augmentation. Every input to the network is scaled such that it has norm 1.*

## 4 Generalization in Deep Networks

In this section we prove formally that  $\rho$ , which is inversely related to the margin, as we discussed, indeed controls a bound on the expected error. We use classical bounds that lead (see Appendix F) to the following theorem

**Observation 3** *With probability  $1 - \delta$*

$$L(f) \leq c_1 \rho \mathbb{R}_N(\tilde{\mathbb{F}}) + c_2 \epsilon(N, \delta) \quad (10)$$

where  $c_1, c_2$  are constants that reflect the Lipschitz constant of the loss function ( for the square loss this requires a bound on  $f(x)$ ) and the architecture of the network. The Rademacher average  $\mathbb{R}_N(\tilde{\mathbb{F}})$  depends on the normalized network architecture and  $N$  (it is independent of the data essentially by considering the worst case). Thus for the same Rademacher complexity, the upper bound for the expected error of the minimizer is smaller for smaller  $\rho$ .

There exist tighter bounds that replace the Rademacher term with estimates that depend on different norms on the weights such as the “path-norm”. An interesting case is in [14? ]. These finer bounds may explain the empirical observation that in the absence of weigh decay the test error of some solution is slightly better than their norm predicts (these cases would depend on different global minima and different sets of weights). The bound supports proves the conjecture in [15] that for deep networks, as for kernel machines, *minimum norm interpolating solutions are the most stable*.

Rho from All Layers

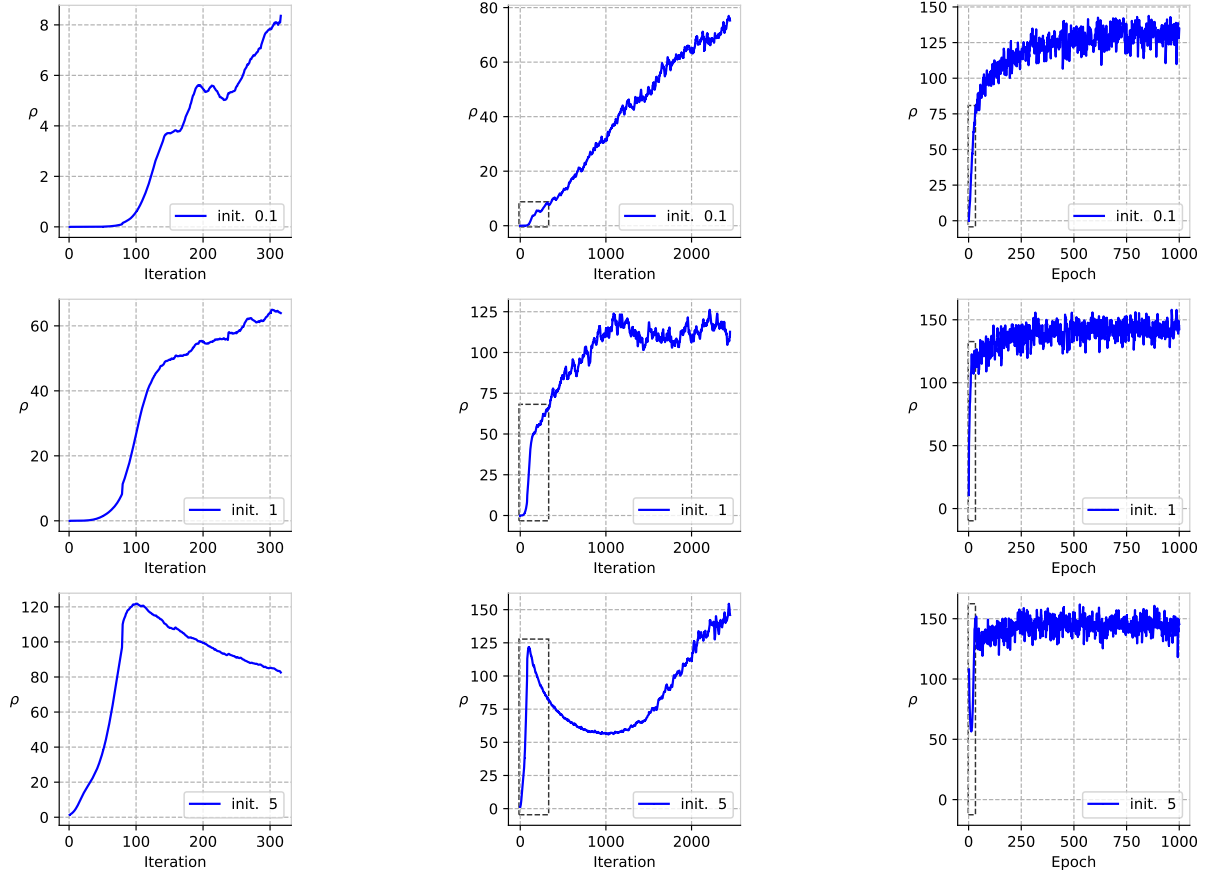


Figure 2: *ConvNet with Batch Normalization and Weight Decay Dynamics of  $\rho$  from experiments in Figure 1. First row: small initialization (0.1). Second row: medium initialization (1). Third row: large initialization (5). A dashed rectangle denotes the previous subplot's domain and range in the new subplot.*

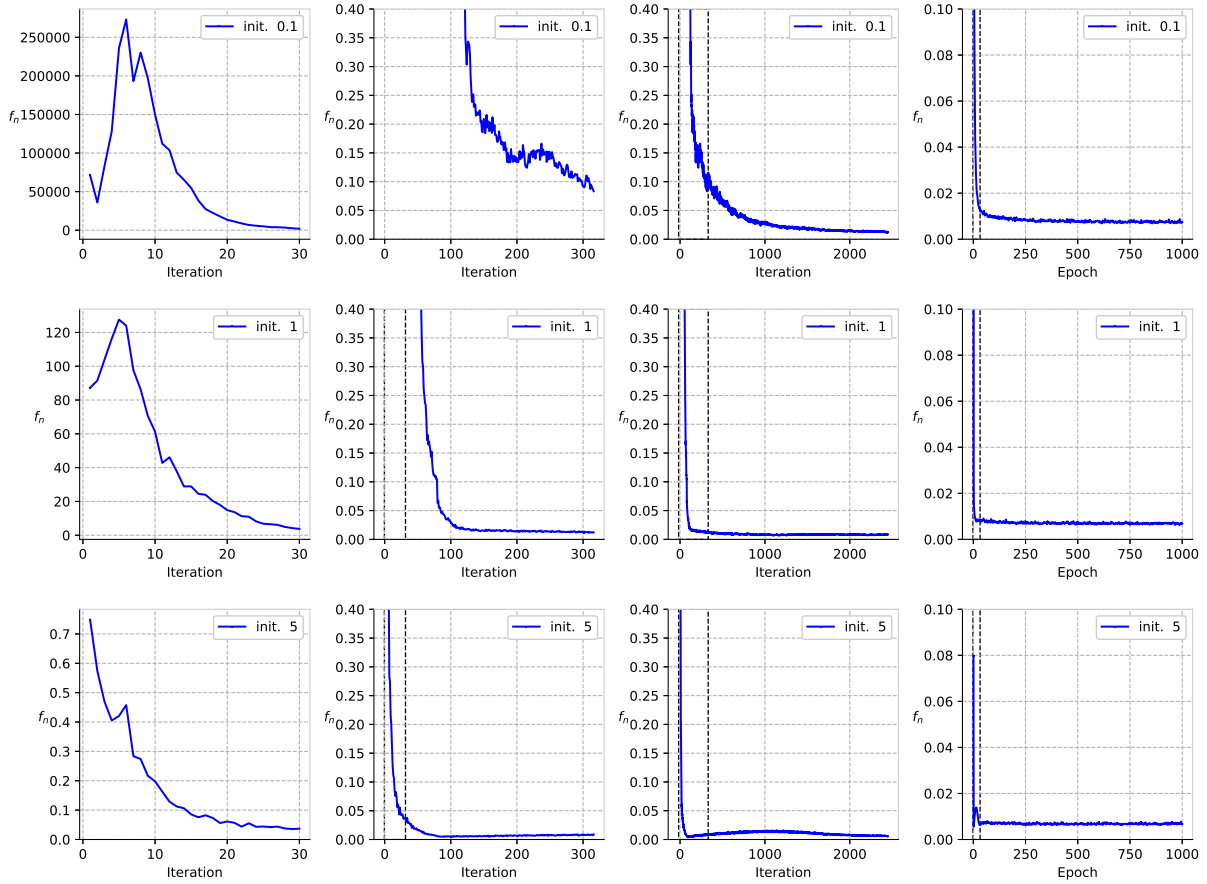


Figure 3: *ConvNet with Batch Normalization and Weight Decay* Dynamics of the average of  $|f_n|$  from experiments in Figure 1. First row: small initialization (0.1). Second row: medium initialization (1). Third row: large initialization (5). A dashed rectangle denotes the previous subplot's domain and range in the new subplot.

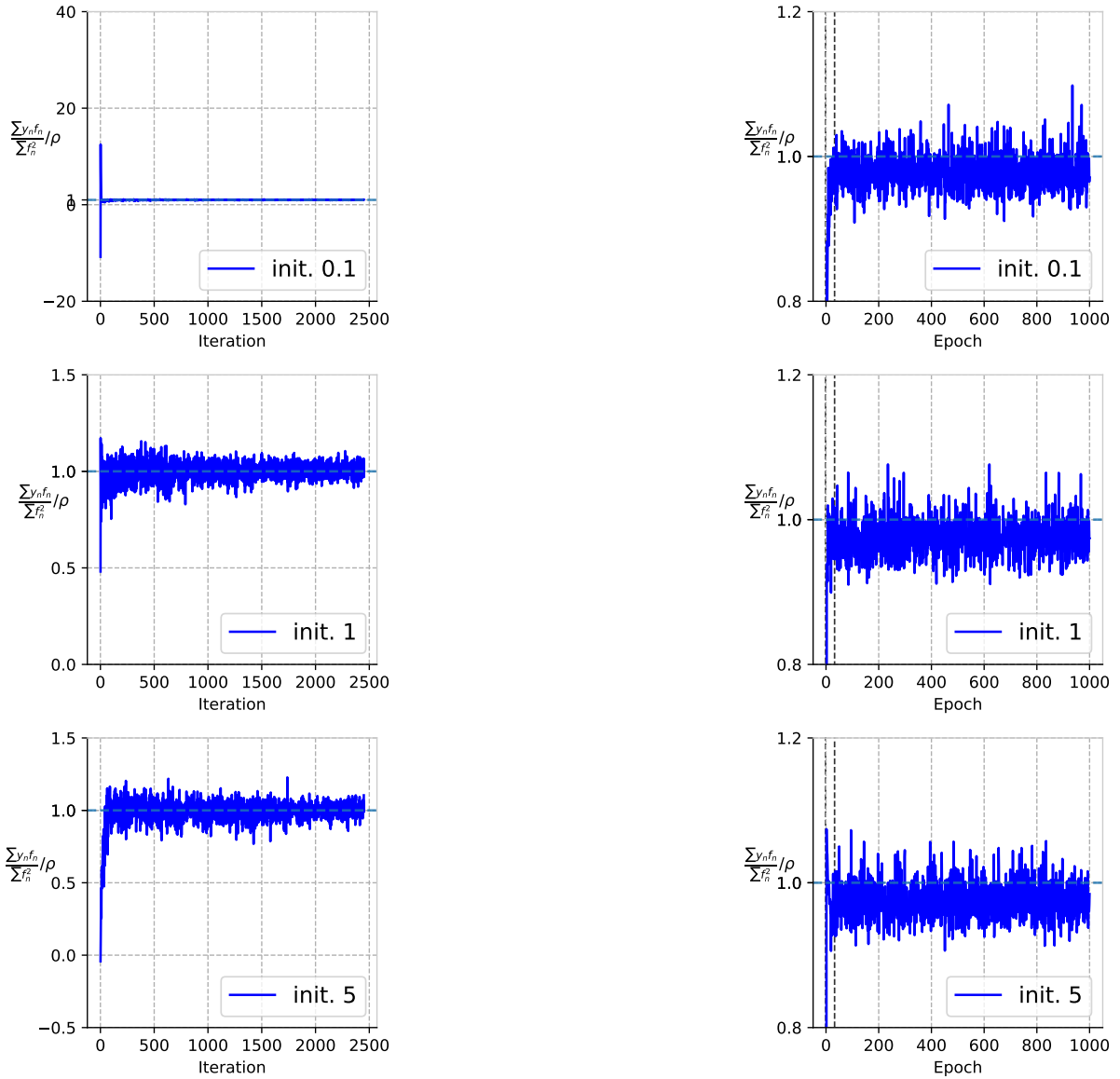


Figure 4: *ConvNet with Batch Normalization and Weight Decay Ratio of  $\frac{\sum y_n f_n}{\sum f_n^2}$  and  $\rho$ . First row: small initialization (0.1). Second row: medium initialization (1). Third row: large initialization (5).*

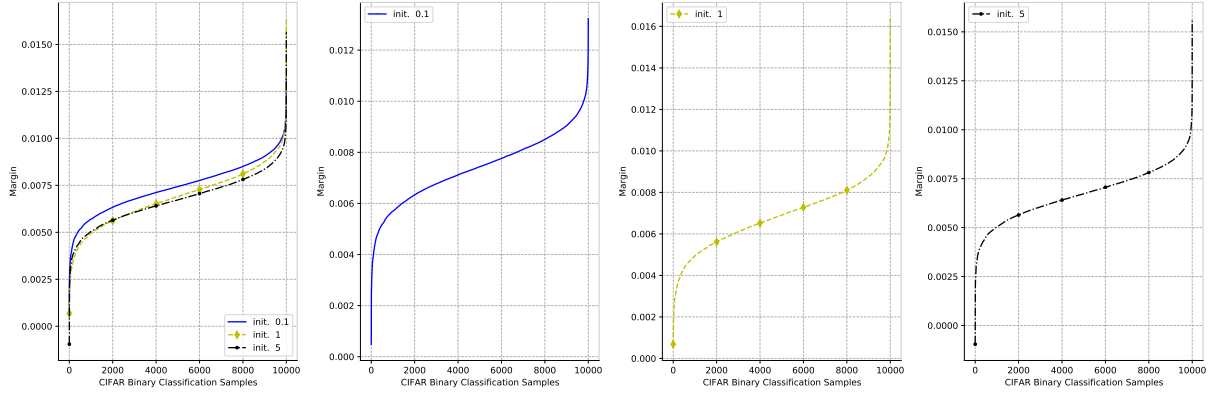


Figure 5: *ConvNet with Batch Normalization and Weight Decay* Margin of all training samples.

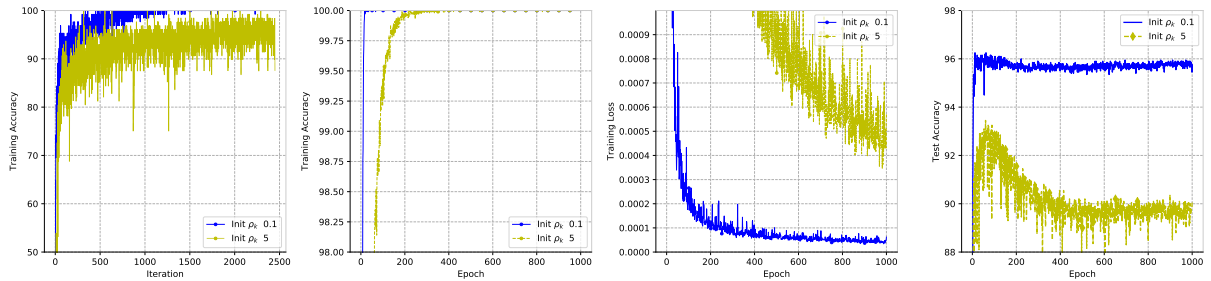


Figure 6: *ConvNet with Batch Normalization but no Weight Decay* Binary classification on two classes from CIFAR-10, trained with MSE loss. The model is a very simple network with 4 layers of convolutions. ReLU nonlinearity is used. Batch normalization is used without parameters (affine=False in PyTorch). 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 either 0.1 or 5. We run SGD with batch size 128, constant learning rate 0.01 and momentum 0.9 for 1000 epochs. No data augmentation. Every input to the network is scaled such that it has norm 1. This is a single run but it is typical for the parameter values we used.

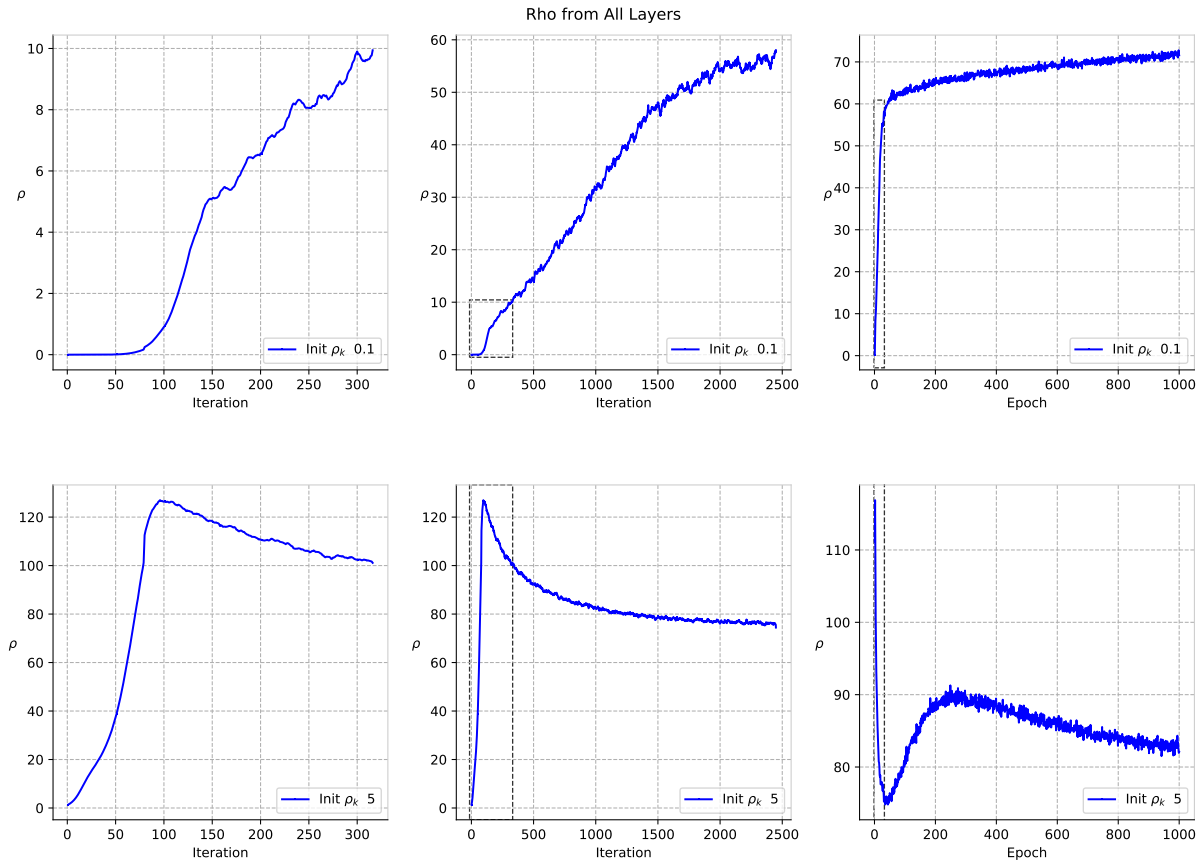


Figure 7: *ConvNet with Batch Normalization but no Weight Decay*. Dynamics of  $\rho$  from experiments in Figure 6. Top row: small initialization (0.1). Bottom row: large initialization (5). The plot starts with  $\rho(0) = 0$  despite an initialization of  $\rho_k = 0$  because the the scaling factor of BN starts from 0. A dashed rectangle denotes the previous subplot's domain and range in the new subplot.

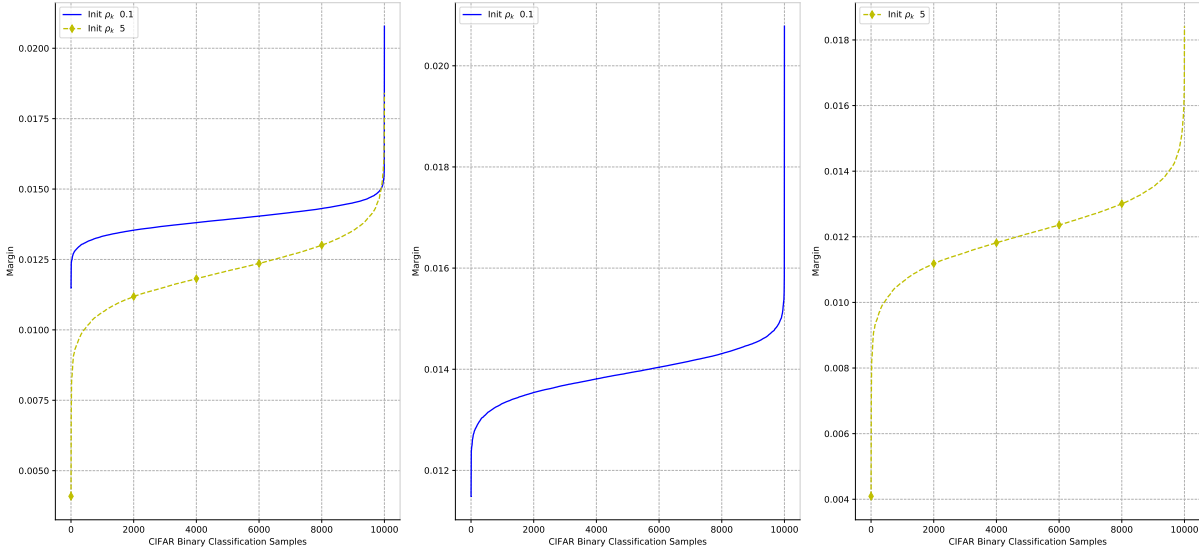


Figure 8: *ConvNet with Batch Normalization but no Weight Decay*. Margin of all training samples (see previous figures). If the solution were to correspond to exactly zero square loss, the margin distribution would be an horizontal line.

## 5 Predictions

- In a recent paper Papyan, Han and Donoho[16] 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 is pushed toward zero. Direct empirical measurements expose an inductive bias they call neural collapse (NC), involving four interconnected phenomena. (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 [NCC] decision rule).

We show here G that these *properties of the Neural Collapse[16] seem to be predicted* by the theory of this paper in the case of binary classification for the global (that is, close-to-zero square-loss) minima, irrespectively of the value of  $\rho_{eq}$ . We recall that the *basic assumptions of the analysis are Batch Normalization and Weight Decay*.

*Neural collapse in the binary classification case.* In a network with  $L$  layers the last layer activations for an input  $x_{i,c}$  where  $c$  is the class – here we consider just two classes  $c = +1$

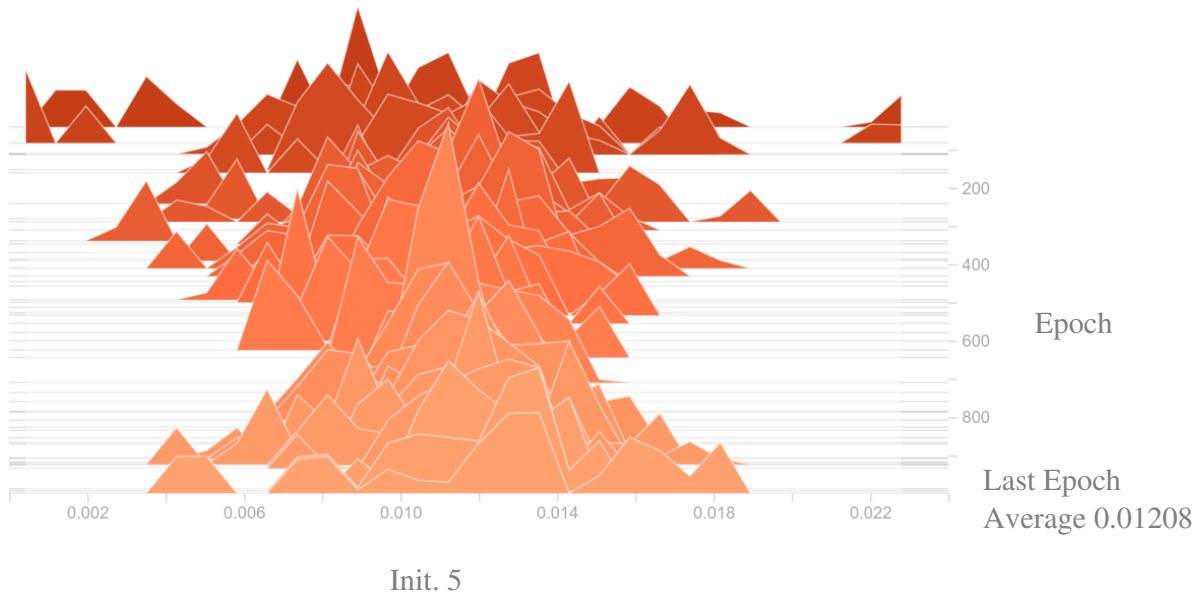
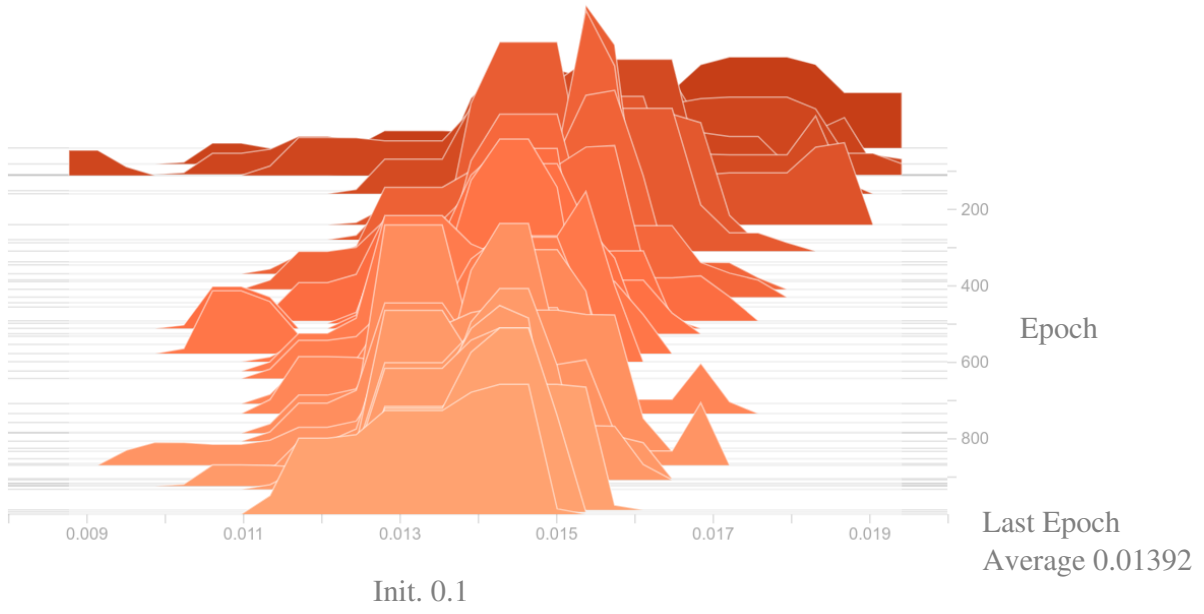


Figure 9: *ConvNet with Batch Normalization but no Weight Decay: Histogram of  $|f_n|$  over time. Top figure: initial  $\rho_k = 0.1$ . Bottom figure: initial  $\rho_k = 5$ .*



or  $c = -1$  – are called  $h_{i,c}$  by [16]. The key observation is that  $h_{i,c} = \frac{\partial f(x_{i,c})}{\partial V_L}$ . Then since  $V_k f_c = \frac{\partial f_c}{\partial V_k}$  at convergence, it follows that  $h^T(x_{i,c}) = V_L f_c$  at convergence. Observe now that

- $V_L f_c$  does not depend on  $i$  implying that the standard deviation of  $h_{i,c}$  wrt  $i$  converges to zero. This is the main part of NC1;
- in the binary case  $h_{+1} \rightarrow V_L^T$  and  $h_{-1} \rightarrow -V_L^T$ . Thus  $\mu_1 = V^T$  and  $\mu_2 = -V^T$ . This is a special case of NC2 (here we do not use hot vectors);
- since  $h(x_{i,c}) = V_L^T f_c$  and  $f_c$  is a number, at convergence  $h$  is proportional to  $V_L^T$ , which is the gist of NC3;
- The result of [17] and NC4 hold. The same is true for our theory (in the previous sections), since it implies the result of [17].

In the Appendix we extend these predictions to the multiclass case. We also show that they hold also in the case of crossentropy, used in [16]. For the exponential loss, in particular, *SGD is required in order to guarantee NC1* and small minibatch sizes are better.

- At a close to zero loss critical point of the flow, Equations 36 become (see Appendix E.3)  $\nabla_{V_k} f(x_j) = V_k f(x_j)$  with  $x_j$  in the training set, which are *powerful constraints* on the weight matrices to which training converges. A specific dependence of the matrix at each layer on matrices at the other layers is thus required. In particular, there are specific relations for each layer matrix  $V_k$  of the type, explained in the Appendix,

$$V_k f = [V_L D_{L-1}(x) V_{L-1} \cdots V_{k+1} D_k(x)]^T D_{k-1}(x) V_{k-1} D_{k-2}(x) \cdots D_1(x) V_1 x, \quad (11)$$

where the  $D$  matrices are diagonal with components either 0 or 1, depending on whether the corresponding RELU unit is on or off.

As described in the Appendix *for linear networks, a class of possible solutions to these constraint equations are projection matrices; another one are orthogonal matrices and more generally orthogonal Stiefel matrices on the sphere*. These are *sufficient but not necessary* conditions to satisfy the constraint equations. The current analysis (in Appendix I) of the constraint equation is quite limited since it holds only for deep linear networks: a full analysis is still missing. Interestingly, randomly initialized weight matrices (an extreme case of the NTK regime) are approximately orthogonal.

## 6 Summary

The main results of the paper analysis can be summarized in the following

**Proposition 1** *If the gradient flow with normalization and weight decay converges to an interpolating solution with near-zero square loss (for  $\lambda > 0$ ), the following properties hold:*

1. For a fixed architecture the global minima with the smallest  $\rho$  are the global minimum norm solutions and have the best margin and the best bound on expected error.
2. Conditions that favour convergence to minimum norm solutions are weight decay  $\lambda$  (with BN), label noise and small initialization (small  $\rho$ ). Weight decay avoids degeneracy in the solution (though the solution may still depend on initial conditions even if the global minima are connected).
3. If there is sufficient overparametrization (width of  $V_1 \geq 2N$  and decreasing width of each layer wrt previous layer) all zero-loss solutions are connected implying that SGD with  $\lambda > 0$  is likely to converge to the minimum norm solutions independently of initial conditions.
4. At the global minima the margins for all training points are equal.
5. The condition  $\frac{\partial f(x_j)}{\partial V_k} = V_k f(x_j)$  which holds at the critical points of the SGD dynamics that are global minima, predicts all properties of the Neural Collapse[16]; it also represents a constraint on the set of weight matrices at convergence.

## 6.1 Remarks

- We assume throughout overparametrization, meaning that there is enough overparametrization to allow for “benign” interpolation of the training data. This requires more parameters (weights) than  $N$  – the number of training data – and in some setting significantly more (see Theorem 5.1 of [13] and earlier comment about it). It also requires that the training points never coincide because this would make interpolation impossible. One expects that if the minimum distance between points is too small, interpolation may become impossible with high probability. A rough bound on the notion of “too close” is provided by the sampling theorem: if two points are closer than the Shannon sampling distance for a function with a cutoff frequency of  $(\frac{2\pi}{T})$  then the samples will be dependent and again make interpolation impossible (in the presence of noise). Following this argument interpolation is possible if the dimensionality of the input is high enough wrt to  $N$ . For convolutional networks the dimensionality is  $\approx (\frac{2\pi}{T})^d$  where  $d$  is the dimensionality of the kernel (for a  $3 \times 3$  kernel with 3 color channels  $d \approx 30$ ) and  $T$  is the sampling distance in the image (or of the “stride”).
- Suppose we control  $\rho_k$  independently of  $V_k$  and of equation 28: can we find  $\rho(t)$  schedules leading more reliably to good solutions independently of initial conditions?
- The role of the Lagrange multiplier term  $\nu \sum_k \|V_k\|^2$  in Equation 3 is different from a standard regularization term because  $\nu$ , determined by the constraint  $\|V_k\| = 1$  can be positive or negative, depending on the sign of the error  $\nu = -\sum_n (\rho^2 f_n^2 - \rho y_n f_n)$ . Thus the  $\nu$  term acts as a regularizer when the norm of  $V_k$  is larger than 1 but has the opposite effect for  $\|V_k\| < 1$ , thus constraining each  $V_k$  to the unit sphere. For the exponential loss

the situation is different and  $\nu$  in  $\nu \sum_k \|V_k\|^2$  acts as a positive regularization parameter, albeit a vanishing one (for  $t \rightarrow \infty$ ).

- For the square loss, convergence of the gradient flow to local minimum norm solutions requires BN or WN and WD, unlike the case of linear networks. For the exponential loss, BN is not needed since minimization of the exponential loss maximizes the margin and minimizes the norm without BN. Thus under the exponential loss, we expect a margin maximization effect for  $t \rightarrow \infty$ , as shown in [3], independently of initial conditions. Deep nets under the square loss are more likely to overfit at long times than under exponential-type loss functions (unless weight decay is used). As a consequence, early stopping is more likely to be effective for the square loss than for exponential-type loss functions. Empirically, it seems that square loss reaches solutions with good test error in multiclass CIFAR10 faster than cross-entropy. Continuing GD, however, sometime yields overfitting for the square loss (and worse test error) but not for cross-entropy. This is interesting because it validates the asymptotic complexity control we described in [18]. It also suggests that in the experiments of [8], early stopping may play a role to obtain results with the square loss case that are as good or better than cross-entropy. We conjecture that the overfitting phenomenon is related to the singular nature of the global critical point when the weight decay  $\lambda$  is zero or too small (see Equations 5 and 6).
- If there exist several almost-interpolating solutions with the same norm  $\rho_{eq}$ , they also have the same margin for each of the training data. Though they have the same norm and the same margin on each of the data point, they may have different ranks of the weight matrices or of the rank of the local Jacobian  $\frac{\partial f_n}{\partial V_k}$  (at the minimum  $W^*$ ). Notice that in deep linear networks the GD dynamics seems to bias the solution towards small rank solutions, since large eigenvalues converge much faster than the small ones [19]. It is unclear whether the rank has a role in our analysis of generalization (it was considered in the first versions of this paper).
- Small initialization ensures that  $\rho$  grows for small values thus exploring first large margin minima – assuming that average separability is reached early, during the first iterations of GD. Why does GD have difficulties in converging in the absence of BN, especially for very deep networks? At the moment, the best answer is that good tuning of the learning rate is important and BN together with weight decay was shown to provide a remarkable autotuning [9]. A closely related observation is E.4.
- The normalization Equation 3 is a precise model of WN. Normalization of the weight matrices  $V_k$  is also an effect of BN. However, *BN is also normalizing each row of each  $V_k$  matrix*, as we mentioned earlier<sup>12</sup>.

---

<sup>12</sup>The normalization may change during training between each training example because of the role of the  $D$  matrices, effectively switching on and off some weights in the network, depending on  $x_n$  and on whether BN is before or after the RELU nonlinearity (as pointed out by A. Banbuski).

- Are there any implications of the theory sketched here for mechanisms of learning in cortex? Somewhat intriguingly, some form of normalization, often described as a balance of excitation and inhibition, has long been thought to be a key function of intracortical circuits in cortical areas. One of the first deep models of visual cortex models, HMAX, explored the biological plausibility of specific normalization circuits with spiking and non-spiking neurons. It is also interesting to note that the Oja rule describing synaptic plasticity in terms of changes to the synaptic weight is the Hebb rule plus a normalization term that corresponds to a Lagrange multiplier.
- The generalization bounds we obtain do NOT necessarily require  $\lambda > 0$  but robust convergence to large margins is helped by  $\lambda > 0$  even with very small  $\lambda$ . The main effect of  $\lambda > 0$  together with very large overparametrization is to make convergence independent of initial conditions.
- The main problems left open by this paper are:
  - The analysis is so far restricted to gradient flow. It should be extended to *gradient descent*.
  - The behavior of gradient descent around the global minima should be analyzed in the limit for  $\lambda \rightarrow 0$ . Equation 6 contains two terms, one reflecting the normalization and the other the regression error. Zero regression error implies that normalization fails at the critical point for  $\lambda = 0$ . It is remarkable that for  $\lambda = 0$  or even more surprisingly for the case of no BN and no WD, the dynamical system still yields good results, provided *initialization is small*. The case of BN+WD is the only one which seems rather independent of initial conditions in our experiments.
  - In this context, an extension of the analysis to SGD may also be critical for providing a satisfactory analysis of convergence.

**Acknowledgments** We are grateful to Shai Shalev-Schwartz, Andrzej Banbuski, Arturo Desza, Akshay Rangamani, Santosh Vempala, David Donoho, Vardan Papyan, X.Y. Han, Silvia Villa and especially to Eran Malach for very useful comments. This material is based upon work supported by the Center for Minds, Brains and Machines (CBMM), funded by NSF STC award CCF-1231216, and part by C-BRIC, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

## References

- [1] Vidya Muthukumar, Adhyayan 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.
- [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] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- [5] Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in Neural Information Processing Systems*, 2016.
- [6] 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.
- [7] 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.
- [8] Like Hui and Mikhail Belkin. Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks, 2020.
- [9] Sanjeev Arora, Zhiyuan Li, and Kaifeng Lyu. Theoretical analysis of auto rate-tuning by batch normalization. *CoRR*, abs/1812.03981, 2018.
- [10] T. Poggio and Y. Cooper. Loss landscape: Sgd has a better view. *CBMM Memo 107*, 2020.
- [11] Tengyuan Liang, Tomaso Poggio, Alexander Rakhlin, and James Stokes. Fisher-rao metric, geometry, and complexity of neural networks. *CoRR*, abs/1711.01530, 2017.
- [12] T. Poggio and Y. Cooper. Loss landscape: Sgd can have a better view than gd. *CBMM memo 107*, 2020.
- [13] 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.
- [14] P. Bartlett, D. J. Foster, and M. Telgarsky. Spectrally-normalized margin bounds for neural networks. *ArXiv e-prints*, June 2017.
- [15] Tomaso Poggio. Stable foundations for learning. *Center for Brains, Minds and Machines (CBMM) Memo No. 103*, 2020.

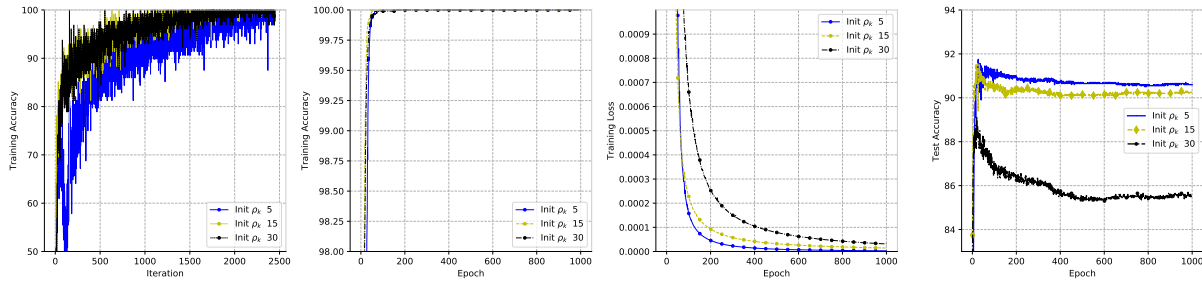


Figure 10: **ConvNet, no Batch Normalization, no Weight Decay.** Binary classification on two classes from CIFAR-10, trained with MSE loss. The model is a very simple network with 4 layers of fully-connected Layers. The ReLU nonlinearity is used. The weight matrices of all layers are initialized with zero-mean normal distribution, scaled by a constant such that the Frobenius norm of each matrix is either 5, 15 or 30. We run SGD with batch size 128, constant learning rate 0.1 and momentum 0.9 for 1000 epochs.. No data augmentation. Every input to the network is scaled such that it has Frobenius norm 1.

- [16] Vardan Pappayan, 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.
- [17] D. Soudry, E. Hoffer, and N. Srebro. The Implicit Bias of Gradient Descent on Separable Data. *ArXiv e-prints*, October 2017.
- [18] T. Poggio, Q. Liao, and A. Banburski. Complexity control by gradient descent in deep networks. *Nature Communication*, 2020.
- [19] Daniel Gissin, Shai Shalev-Shwartz, and Amit Daniely. The Implicit Bias of Depth: How Incremental Learning Drives Generalization. *arXiv e-prints*, page arXiv:1909.12051, September 2019.
- [20] Jeff Z. HaoChen, Colin Wei, Jason D. Lee, and Tengyu Ma. Shape matters: Understanding the implicit bias of the noise covariance, 2020.
- [21] Paulo Jorge S. G. Ferreira. The existence and uniqueness of the minimum norm solution to certain linear and nonlinear problems. *Signal Processing*, 55:137–139, 1996.
- [22] O. Bousquet, S. Boucheron, and G. Lugosi. Introduction to statistical learning theory. pages 169–207, 2003.
- [23] Daniel Kunin, Jonathan M. Bloom, Aleksandrina Goeva, and Cotton Seed. Loss landscapes of regularized linear autoencoders. *CoRR*, abs/1901.08168, 2019.
- [24] Kui Jia, Shuai Li, Yuxin Wen, Tongliang Liu, and Dacheng Tao. Orthogonal deep neural networks. *CoRR*, abs/1905.05929, 2019.

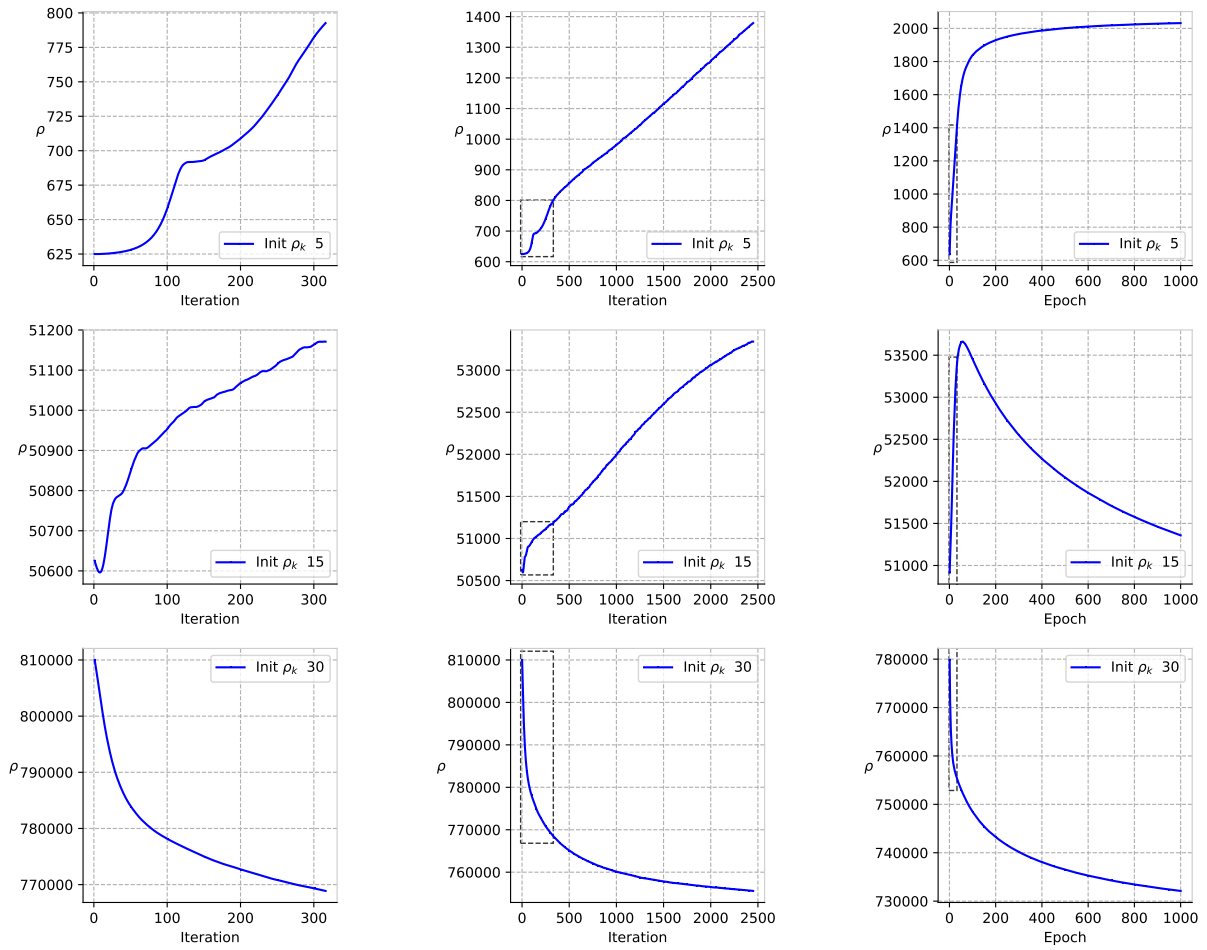


Figure 11: *ConvNet, no Batch Normalization, no Weight Decay*. Dynamics of  $\rho$  from experiments in Figure 10. First row: small initialization (5). Second row: large initialization (15). Third row: extra large initialization (30). A dashed rectangle denotes the previous subplot's domain and range in the new subplot. More details to be added.

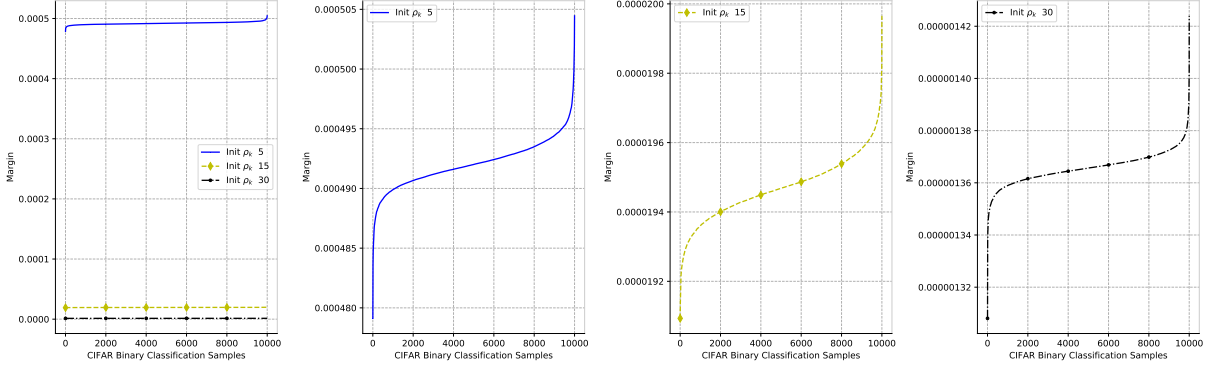


Figure 12: *ConvNet, no Batch Normalization, no Weight Decay.* Margin of all training samples

## A Normalization during Gradient Descent

### A.1 Weight Normalization

For each layer (for simplicity of notation and consistency with the original weight normalization paper), weight normalization [5] defines  $v$  and  $g$  in terms of  $w = g \frac{v}{|v|}$ . The dynamics on  $g$  and  $v$  is induced by the gradient dynamics of  $w$  as follows (assuming  $\dot{w} = -\frac{\partial L}{\partial w}$ )

$$\dot{g} = \frac{v^T}{\|v\|} \dot{w} \quad (12)$$

and

$$\dot{v} = \frac{g}{\|v\|} S \dot{w} \quad (13)$$

with  $S = I - \frac{vv^T}{\|v\|^2}$ .

We claim that this is the same dynamics obtained from tangent gradient for  $p = 2$ . In fact, compute the flows in  $\rho, v$  from  $w = \rho v$  as

$$\dot{\rho} = \frac{\partial w}{\partial \rho} \frac{\partial L}{\partial w} = v^T \dot{w} \quad (14)$$

and

$$\dot{v} = S \rho \dot{w} \quad (15)$$

Clearly the dynamics of this algorithm is the same as standard weight normalization if  $\|v\|_2 = 1$ , because then Equations 12 and 13 become identical to Equations 14 and 15 with  $g$  corresponding to  $\rho$ . We now observe, multiplying Equation 13 by  $v^T$ , that  $v^T \dot{v} = 0$  because  $v^T S = 0$ , implying that  $\|v\|^2$  is constant in time. Thus if  $\|v\| = 1$  at initialization, it will not



change (at least in the noiseless case). Thus *the dynamics of Equations 12 and 13 is the same dynamics as Equations 14 and 15*. It is also easy to see that the dynamics above is not equivalent to the standard dynamics on the  $w$  (see [7]).

## A.2 Batch Normalization

Batch normalization [4] for unit  $i$  in the network normalizes the input vector of activities to unit  $i$  – that is it normalizes  $X^j = \sum_j W^{i,j} x_j$ , where  $x_j$  are the activities of the previous layer. Then it sets the activity to be

$$Y^j = \gamma \cdot \hat{X}^j + \beta = \gamma \frac{X^j - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta,$$

where  $\gamma, \beta$  are learned subsequently in the optimization and

$$\mu_B = \frac{1}{N} \sum_{n=1}^N X_n \quad \sigma_B^2 = \frac{1}{N} \sum_{n=1}^N (X_n - \mu_B)^2.$$

Note that both  $\mu_B$  and  $\sigma_B^2$  are vectors, so the division by  $\sqrt{\sigma_B^2 + \epsilon}$  has to be understood as a point-wise Hadamard product  $\odot (\sigma_B^2 + \epsilon)^{-1/2}$ . The gradient is taken wrt the new activations defined by the transformation above.

Unlike Weight Normalization, the Batch Normalization equations do not include an explicit computation of the partial derivatives of  $L$  with respect to the new variables in terms of the standard gradient  $\frac{\partial L}{\partial w}$ . The reason is that Batch Normalization works on an augmented network: a BN module is added to the network and partial derivatives of  $L$  with respect to the new variables are directly computed on its output. Thus the BN algorithm uses only the derivative of  $L$  wrt the old variables as a function of the derivatives of  $L$  wrt new variables in order to update the parameters below the BN module by applying the chain rule. Thus we have to estimate what BN *implies* about the partial derivatives of  $L$  with the respect to the new variables as a function of the standard gradient  $\frac{\partial L}{\partial w}$ .

To see the nature of the dynamics implied by batch normalization we simplify the original Equations (in the Algorithm 1 box in [4]). Neglecting  $\mu_B$  and  $\beta$  and  $\gamma$ , we consider the core transformation as  $\hat{X} = \frac{X}{\sigma_B}$  which, assuming fixed inputs, becomes  $\hat{X} = \frac{X}{|X|}$  which is mathematically identical with the transformation considered in [7]. In a similar way the dynamics of  $w = \frac{\partial L}{\partial w}$  induces the following dynamics on  $\hat{X}$ :

$$\dot{\hat{X}} = \frac{\partial \hat{X}}{\partial X} \dot{X} \tag{16}$$

where  $\dot{x} = \nabla_x L$ . We consider  $X \in \mathbb{R}^{N \times D}$ . In the  $D = 1$  case, we get

$$\frac{\partial \hat{X}}{\partial X} = (\sigma_B^2 + \epsilon)^{-1/2} \left[ -\frac{1}{N} \hat{X} \hat{X}^T + I \right].$$

In the general  $D$ -dimensional vector case, this generalizes to

$$\frac{\partial \hat{X}}{\partial X} = (\sigma_B^2 + \epsilon)^{-1/2} \left[ -\frac{1}{N} \hat{X}^T \odot \hat{X} + I \right].$$

Notice that  $I - \hat{X} \hat{X}^T = S$ . Since  $x = W x_{input}$  this shows that batch normalization is closely related to *gradient descent algorithms with unit  $L_2$  norm constraint of the tangent gradient type*. Because of the simplifications we made, there are other differences between BN and weight normalization, some of which are described in the remarks below.

*Remarks*

1. Batch normalization, does not control directly the norms of  $W_1, W_2, \dots, W_K$  as WN does. Instead it controls the norms

$$\|x\|, \|\sigma(W_1 x)\|, \|\sigma(W_2 \sigma(W_1 x))\|, \dots \quad (17)$$

2. In the multilayer case, BN controls separately the norms  $\|V_i\|$  of the weights into unit  $i$ , instead of controlling the overall Frobenius norm of the matrix of weights as WN does. Of course control of the  $\|V_i\|$  implies control of  $\|V\|$  since  $\|V\|^2 = \sum_i \|V_i\|^2$ .

## B Gradient flow for $\rho$ and $V_k$

Gradient descent on  $\mathcal{L} = \frac{1}{N} (\sum_n g_n^2 - 2 \sum_n y_n g_n + N)$  (using  $g_n = g(x_n)$ ) gives

$$\dot{W}_k = -\frac{2}{N} \sum_n (g_n - y_n) \frac{\partial g_n}{\partial W_k} \quad (18)$$

that is

$$\dot{W}_k = -\frac{\partial \mathcal{L}}{\partial W_k} = -\frac{2}{N} \sum_n g_n \frac{\partial g_n}{\partial W_k} + \frac{2}{N} \sum_n y_n \frac{\partial g_n}{\partial W_k} \quad (19)$$

We now derive the dynamics of the norm and of the normalized weights. We define  $g(x) = \rho f(x)$ .  $\rho$  is the product of the Frobenius norms of the weight matrices of the  $L$  layers in the network.  $f$  is the corresponding network with normalized weight matrices (because the ReLU is homogeneous [7]). In the following we use the notation  $f_n$  meaning  $f(x_n)$ . We also assume  $\|x\| = 1$  implying  $\|f(x)\| \leq 1$  at convergence;

### B.1 Dynamics under normalization

Gradient flow on  $\mathcal{L} = \sum_n (\rho f_n - y_n)^2 + \nu \sum_k \|V_k\|^2 + \lambda \rho^2$  with  $\|V_k\|^2 = 1$  is completely equivalent (for  $\lambda = 0$ ) to ‘‘Weight Normalization’’ [7] for deep networks.

Assuming that  $\rho_k = 1, \forall k < L$  and  $\rho_L = \rho$ , gradient flow on  $L$  wrt  $\rho$  gives,

$$\dot{\rho} = -2 \sum_n \ell_n f_n - 2\lambda\rho \quad (20)$$

Gradient flow on  $L$  wrt  $V_k$  gives

$$\dot{V}_k = -\frac{\partial L}{\partial V_k} = -2 \sum_n (\rho f_n - y_n) \rho \frac{\partial f_n}{\partial V_k} - 2\nu V_k. \quad (21)$$

Because of the constraint imposed via Lagrange multipliers  $\|V_k\|^2 = 1$ ,  $V_k^T \dot{V}_k = 0$ , which gives  $\nu = -\sum_n (\rho^2 f_n^2 - \rho y_n f_n)$ .

In summary, gradient flow on  $L$  wrt  $\rho$  and  $V_k$  gives

$$\dot{\rho} = -2[\sum_n \rho(f_n)^2 - \sum_n f_n y_n] - 2\lambda\rho = -2 \sum_n \ell_n f_n - 2\lambda\rho. \quad (22)$$

where  $\ell_n = \rho f_n - y_n$  and

$$\dot{V}_k = 2 \sum_n [(\rho f_n - y_n) \rho (-\frac{\partial f_n}{\partial V_k}) + 2V_k \rho f_n (\rho f_n - y_n)] = 2\rho \sum_n [(\rho f_n - y_n)(V_k f_n - \frac{\partial f_n}{\partial V_k})] \quad (23)$$

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  $\frac{\partial g_n(W)}{\partial W_k} = \rho \frac{\partial f_n(V)}{\partial V_k}$ ,  $\forall k < L$  and  $\frac{\partial g_n(W)}{\partial W_L} = \frac{\partial f_n(V)}{\partial V_k}$ .

This dynamics – where there is a “vanishing” Lagrange multiplier  $\nu$  – 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 [9].

Notice that  $\dot{\rho} = 0$  if  $y_n f_n = 1$  and  $\rho_k = 1$ ;  $\rho_{eq}$  is a critical point for the dynamics of  $\rho$  under GD. In the case of SGD the asymptotic value of  $\rho$  for fixed  $\sum f_i y_i$  may fluctuate randomly around the  $\frac{\sum_n y_n f_n}{\sum_n f_n^2}$ . Furthermore, the lowest possible value of  $\rho_k$  at equilibrium ( $\dot{\rho}_k = 0$ ) is  $\rho_k = 1$  which can be achieved if  $y_n f_n$  is either  $= 1$  or  $= 0$ . Values  $y_n f_n = 1, \rho = 1$  are stationary points of the dynamics of  $V_k$  given by  $\dot{V}_k = 0$ : they are minimizers with zero square loss.

## C Maximum margin and minimum norm

**Lemma 2** *The maximizer of the margin under the constraint  $\|V_k\| = 1$  is the minimum norm solution under the constraint  $y_n f_n \geq 1$ ,  $\forall n$ .*

Minimum norm regression of binary labels is

$$\min_{W_k} \frac{1}{2} \|W_k\|^2, \quad \forall k \text{ subj. to } y_i f(W_k, \dots, W_1; x_i) = 1, \quad i = 1, \dots, N. \quad (24)$$

Minimum norm binary classification is

$$\min_{W_k} \frac{1}{2} \|W_k\|^2, \quad \forall k \text{ subj. to } y_i f(W_k, \dots, W_1; x_i) \geq 1, \quad i = 1, \dots, N. \quad (25)$$

Clearly classification involves minimizing over a larger class of functions than regression. The result will be in general different.

**Observation 4** *Minimum norm binary classification under the square loss with margin 1 (implying  $f_V(x_i) \geq 1 \forall i$ ) is not (in general) interpolation of all the data.*

Notice that hard margin SVM is a case in point: the SVs interpolate their data point, but other non-support vectors may have margin greater than one. This indicates that there should be better algorithm to train deep networks for classification than regression.

## D Unnormalized GD

Here we assume gradient flow without BN (and without weight decay), assuming, for simplicity, that at initialization all the layers have the same norm, that is  $\rho_k$  is the same for all  $k$  at initialization. Because of this assumption we can use the following

**Lemma 3**  $\frac{\partial \rho_k^2}{\partial t}$  is independent of  $k$ .

to claim that all  $\rho_k$  are the same at all times. Thus  $\rho = \rho_k^L$ , where  $L$  is the number of layers.

*Proof* Consider  $\frac{\partial \|W_k\|^2}{\partial t}$ . The calculation follows the case for the exponential loss:

$$\frac{\partial \|W_k\|^2}{\partial t} = 2W_k \frac{\partial W_k}{\partial t} = \frac{4}{N} \sum_n g_n^2 + \frac{4}{N} \sum_n g_n \quad (26)$$

because of the structural lemma. Thus the time evolution of  $\rho^2 = \|W_K\|^2$  is independent of  $k$ .

Then we obtain the dynamical system

$$\dot{\rho}_k = -\frac{\partial L}{\partial \rho_k} = -2L \sum_n (\rho_k^L f_n - y_n) f_n \rho_k^{L-1} = -2\rho_k^{L-1} [\sum_n \rho_k^L (f_n)^2 - \sum_n f_n y_n] \quad (27)$$

which can be rewritten in terms of  $\rho = \rho_k^L$  using  $\dot{\rho} = \sum_k \frac{\partial \rho}{\partial \rho_k} \dot{\rho}_k$  as

$$\dot{\rho} = 2L\rho^{\frac{2L-2}{L}} [\sum_n f_n y_n - \sum_n \rho (f_n)^2] \quad (28)$$

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 simulations in the appendix (look at  $\rho$  for small initialization during the first 50 or so iterations).

The dynamics of Equation 28 is that the smaller  $\rho_{t=0}$  is, the longer it takes to  $\rho$  to grow (this phenomenon increases with increasing number of layers  $L$ ). Thus  $\rho$  is constrained by the nonlinear dynamics to be very small for a transient phase  $T$  of GD iterations (as we mentioned,  $T$  is longer with more layers and longer with smaller initialization) and then to grow slowly while  $f_n$  grows towards 1 (implying that  $\sum f_n^2$  approaches  $\sum y_n f_n$ ). Part of this dynamics was analyzed by Shalev-Schwartz [19] in a different context.

If the initial conditions are  $\rho_{t=0} \approx 0$ ,  $\rho(t)$  will eventually grow (most of the time, when it does not go to zero), but slowly for a longish time. Part of this behavior can be explained by the logistic equation in which the coefficients change with time, with  $y_n f_n$  decreasing slowly. As a consequence, the rate of increase of  $\rho$  decreases, though the asymptotic value of  $\rho = \frac{\sum y_n f_n}{\sum f_n^2}$  increases. until a critical point of the flow is reached

Compare this with the case in which  $\rho$  is large at initialization: then  $\rho$  may decrease until a critical point is reached. As we already noticed, there are plenty of critical points at large  $\rho$  (stationary points for  $\rho$  and  $V_k$ ) because under appropriate overparametrization, almost every perturbations of the weights before the linear classifier at the top yields a different interpolating solution. Of course, in the presence of significant weight decay the gradient flow may escape these minima.

## E Dynamics and equilibria for $V_k$ and $\rho$

When  $\lambda > 0$  the terms  $\ell_n = (\rho f_n - y_n) \neq 0$  in Equation 8. It is then reasonable to assume that  $\alpha_n > 0$  and that  $V^{eq}$  at the minimum can be written as

$$V_k^{eq} = \sum \alpha_n \frac{\partial f_n}{\partial V_k} \quad (29)$$

where  $\alpha_n = \frac{\rho f_n - y_n}{\sum_n (\rho f_n - y_n) f_n}$ <sup>13</sup>.

### E.1 Weight decay and label noise

Adding a term  $\lambda \|W_k\|^2$  to the Lagrangian corresponding to weight decay changes the dynamics of  $\rho$  but not the dynamics of  $V_k$ . The Equation for  $\rho$  becomes (with BN and WD) Equation 7, that is

$$\dot{\rho} = -2 \left[ \sum_n \rho (f_n)^2 - \sum_n f_n y_n \right] - 2\lambda \rho \quad (30)$$

which has an equilibrium given by

$$\rho_{eq} = \frac{\sum_n y_n f_n}{\lambda + \sum_n f_n^2}. \quad (31)$$

Notice that label noise (adding  $\pm\delta$  – with  $\delta$  a small random real number to the labels), as suggested by Jason Lee and coworkers[20], may play a role somewhat similar to a regularization

---

<sup>13</sup>

Notice that in general not all of the  $N$  terms in Equation 29 are different from zero<sup>14</sup> or independent of each other (an upper bound is set by the rank of Jacobian  $\frac{\partial f_n}{\partial V_k}$ ). As an example consider the degenerate linear case when  $f$  is a linear function.

$\lambda$  by eliminating interpolation<sup>15</sup>. Both label noise and weight decay introduce a bias towards small  $\rho_{eq}$  minima – and thus better generalization – even for large initializations.

## E.2 Convergence of Linear Networks with Normalization and Regularization

Consider the separable case of a linear network ( $f(x) = \rho v^T x$ ). The dynamics is – with  $\ell_n = e^{-\rho y_n v^T x_n}$  for the exponential loss and  $\ell_n = (\rho v^T x_n - y_n)$  for the square loss–

$$\dot{\rho} = \frac{2}{\rho} \sum_{n=1}^N \ell_n y_n v^T x_n - 2\lambda\rho \quad (32)$$

and

$$\dot{v} \propto \sum_{n=1}^N \ell_n (x_n - v v^T x_n). \quad (33)$$

If  $\lambda > 0$  it is reasonable to assume  $\sum_{n=1}^N \ell_n y_n \neq 0$ . Thus

$$\dot{v} \propto \sum_{j=1}^N \alpha_j (I - v v^T) x_j. \quad (34)$$

If gradient flow converges to  $\dot{v} = 0$ , the solution  $v$  must satisfy  $v v^T x = x$ , where  $x = \sum_{j=1}^N \alpha_j x_j$ . Assume  $\|x\| = 1$ . Then  $v = x$ . Since the operator  $T$  in  $v(t+1) = T v(t)$  associated with equation 34 is not expanding [21] (because  $v$  has unit norm), there is a fixed point  $v = x$  which is *independent of initial conditions*. Numerical simulation show that this is not true for  $\lambda = 0$ .

## E.3 Convergence of Networks with Normalization

Consider

$$\dot{V}_k = \alpha (I - V_k V_k^T) \sum_{n=1}^N (y_n - \rho f_n) \frac{\partial f_n}{\partial V_k}. \quad (35)$$

Let us define  $\alpha_n = (y_n - \rho f_n)$  and an “average”  $\frac{\hat{\partial} f}{\partial V_k} = \sum_j \alpha_j \frac{\partial f_j}{\partial V_k}$ . If  $\lambda > 0$  then  $\alpha_n \neq 0, \forall n$ . Then  $(I - V_k V_k^T) \frac{\hat{\partial} f}{\partial V_k} = \dot{V}_k$ . If there is convergence, that is  $\dot{V}_k = 0$ , then it makes sense to assume that in most cases

$$V_k \hat{f} = \frac{\partial f}{\partial V_k} \quad (36)$$

with  $\hat{f} = V^T \frac{\partial f}{\partial V_k}$ <sup>16</sup>.

<sup>15</sup>Label noise also makes  $\sum_n f_n y_n$  smaller, decreasing the equilibrium  $\rho$  and biasing the final solution to have larger margin

<sup>16</sup>If  $V_k^T \frac{\partial f_n}{\partial V_k} = f_n$  with the same  $V_k$  for all  $n$ , then  $\hat{f} = \sum_j \alpha_j f_j$ . The equation provides constraints on the weights  $V_k$  and the other layer weights at convergence.

$$(\rho f_n - y_n) \frac{\partial f_n}{\partial V_k} = (\rho f_n - y_n)(V_k^{eq} f_n), \quad \forall n = 1, \dots, N. \quad (37)$$

Thus, assuming  $\lambda > 0$ , in most cases at equilibrium the following holds

$$V_k^{eq} = \sum \alpha_n \frac{\partial f_n}{\partial V_k}, \quad (38)$$

with  $\alpha_n = \frac{\rho f_n - y_n}{\sum (\rho f_n - y_n) f_n}$ . SGD with minibatches of size 1 (the argument can be extended to other sizes  $< N$ ) has stationary points given by [10]

$$0 = (I - V_k V_k^T) \ell(x_n) \frac{\partial f(x_n)}{\partial V_k}, \quad \forall n. \quad (39)$$

which implies (for  $\lambda > 0$ )

$$\frac{\partial f(x_n)}{\partial V_k} = V_k f(x_n), \quad \forall n. \quad (40)$$

#### E.4 Dynamics with and without normalization

The dynamics with normalization is

$$\dot{\rho} = -2 \sum_n \ell_n f_n - 2\lambda\rho. \quad (41)$$

and

$$\dot{V}_k = 2\rho \sum_n [(\rho f_n - y_n)(V_k f_n - \frac{\partial f_n}{\partial V_k})] \quad (42)$$

The dynamics without normalization is

$$\dot{\rho} = -2 \sum_n \ell_n f_n - 2\lambda\rho. \quad (43)$$

and

$$\dot{V}_k = \frac{2}{\rho} \sum_n [(\rho f_n - y_n)(V_k f_n - \frac{\partial f_n}{\partial V_k})] \quad (44)$$

The two dynamics have the same critical points but are different. Recall that at convergence  $\rho_{eq}$  corresponds to the inverse of the margin  $f^* = f_n$  which is the same for all  $n$  when both normalization and weight decay are present. Thus  $\dot{V}_k, k < L$  is proportional to the inverse of the margin in the normalized case and directly proportional to the margin in the unnormalized case. The factor  $\frac{1}{f^*}$  or  $f^*$  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 better strategy than the opposite, since large margin corresponds to "good" minima in terms of generalization (for classification).

In this context it is interesting to compute the dynamics of  $W_k^{normal}$  and  $W_k$  induced respectively by the two dynamics above in terms of the unnormalized dynamics  $\dot{W}_k$ . The result is

$$\dot{W}_k = V_k \dot{\rho} + \dot{V}_k \rho = V_k V_k^T \dot{W}_k + S \dot{W}_k = \dot{W}_k \quad (45)$$

for the un-normalized case and

$$\dot{W}_k^{normal} = (\rho^2 + (I - \rho^2) V_k V_k^T) \dot{W}_k \quad (46)$$

for the normalized case.

Clearly the dynamics and, therefore the trajectories, in the two cases are different though the critical points are the same.

## F Margins, $\rho$ and expected error

Assuming that weight decay, small initialization and  $\lambda > 0$  provide a bias towards solution with “large” margin, the next step is to use simple bounds [22] to claim better expected error (and better stability) for those solutions.

A typical generalization bound that holds with probability at least  $(1 - \delta)$ ,  $\forall g \in \mathbb{G}$  has the form [22]:

$$|L(g) - \hat{L}(g)| \leq c_1 \mathbb{R}_N(\mathbb{G}) + c_2 \sqrt{\frac{\ln(\frac{1}{\delta})}{2N}} \quad (47)$$

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;  $c_1, c_2$  are constants that reflect 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

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

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 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  $\rho$ s. 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 the observation that, because of homogeneity of the networks, the empirical Rademacher complexity satisfies the property,

$$\mathbb{R}_N(\mathbb{G}) = \rho \mathbb{R}_N(\mathbb{F}), \quad (48)$$

where  $\mathbb{G}$  is the space of functions of our unnormalized networks and  $\mathbb{F}$  denotes the corresponding normalized networks<sup>17</sup>. This observation allows us to use the bound Equation 47 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

$$L_0(f^a) \leq A\rho_a + \epsilon \quad (49)$$

and

$$L_0(f^b) \leq A\rho_b + \epsilon \quad (50)$$

*Thus the bound for the expected error  $L_0(f^a)$  is better than the bound for  $L_0(f^b)$ .*

Similar results can be obtained taking into account different  $\hat{L}(f)$  for the normalized  $f^a$  and  $f^b$  under different  $\gamma$  in Equation 47, that is

$$|L(f) - \hat{L}(f)| \leq c_1 \mathbb{R}_N(\tilde{\mathbb{F}}) + c_2 \sqrt{\frac{\ln(\frac{1}{\delta})}{2N}}. \quad (51)$$

The Rademacher term can be replaced by tighter complexity estimates. An example is provided in [14] which gives a spectral complexity bound for neural networks that depends on the spectral norm and  $(2, 1)$  norm of the weight matrices.

## G Towards Predicting NC1 to NC4 (with A. Banburski and A. Rangamani)

In a recent paper Papyan, Han and Donoho[16] 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 is pushed toward zero. Direct empirical measurements expose an inductive bias they call neural collapse (NC), involving four interconnected phenomena. (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

---

<sup>17</sup>Furthermore, the Rademacher complexity of the space of functions associated with normalized networks of the same architecture is the same (see [14]).

whichever class has the closest train class mean (i.e., the nearest class center [NCC] decision rule).

We show here that these *properties of the Neural Collapse*[16] are predicted for the case of *binary classification* by the theory of this paper. In a network with  $L$  layers the last layer unit activations for an input  $x_{i,c}$  where  $c$  is the class are a vector  $h_{i,c} \in \mathbf{R}^p$  by [16]. To conform to their notation we consider in this section a slightly different network from the one considered in the paper. Until now, we assumed that the network has one scalar output which is ideally  $\pm 1$ . Here we consider instead a network with  $C$  outputs (in the binary case  $C = 2$ ; one output represents the positive class and the other the negative one, both trained to take the value  $+1$  for the respective class). The last layer weights are a matrix  $V_L \in \mathbf{R}^{C,p}$ <sup>18</sup>. The quantities considered by [16] are  $\mu_c = \frac{1}{CN} \sum_i^{CN} h_{i,c}$ ,  $\mu_c \in \mathbf{R}^p$ , where  $N$  is the number of training examples in each class) and  $\mu_G = \frac{1}{C} \sum_c^C \mu_c$ ,  $\mu_G \in \mathbf{R}^p$ . Furthermore  $\tilde{\mu}_c = \frac{(\mu_c - \mu_G)}{\|\mu_c - \mu_G\|}$ . From now on we call  $V_L$  simply  $V^T$  for simplicity of notation. We observe that  $V_c^T h_{i,c'} = f_c(x_{i,c'})$ , since the network is in our new notation  $f_c(x_{i,c'}) = (V_c^T \sigma(V_{L-1} \cdots \sigma(V_1 x_{i,c'}))) = V_c^T h_{c',i}$ .

## G.1 Square loss

**Corollary 4** *Under the assumptions of square loss with Batch Normalization and Weight Decay, the gradient flow equations imply NC1, NC2, NC3, NC4.*

*Proof*

Consider a ReLU deep network  $f_W(x) = W_L \sigma(W_{L-1} \cdots W_2 \sigma(W_1 x) \cdots)$ , that is trained on a dataset  $S = \{(x_n, y_n)\}$  that is a  $C$ -way classification problem. Here we will use  $y_n$  to denote both the class labels and the one hot vectors indicating the class labels. We will use the  $L_2$  regularized square loss as our training objective. The loss is thus:

$$\mathcal{L}(W) = \frac{1}{2} \sum_n \sum_{i=1}^C (y_n^{(i)} - f_W^{(i)}(x_n))^2 + \frac{\lambda}{2} \sum_l \|W_l\|_F^2 \quad (52)$$

We use gradient flow to train the network.  $\dot{W} = -\frac{\partial \mathcal{L}}{\partial W}$ . Let us analyze the dynamics of the last layer, considering each row  $W_L^c$  of  $W_L$  separately:

$$\begin{aligned} \dot{W}_L^c &= \sum_n (y_n^c - \langle W_L^c, h(x_n) \rangle) h(x_n) - \lambda W_L^c \\ &= \sum_{n \in N(c)} (1 - \langle W_L^c, h(x_{n(c)}) \rangle) h(x_{n(c)}) + \sum_{n \in N(c'), c' \neq c} (-\langle W_L^c, h(x_{n(c')}) \rangle) h(x_{n(c')}) - \lambda W_L^c \end{aligned} \quad (53)$$

Let us assume that at equilibrium we achieve quasi-interpolation, with  $f_W^{(c)}(x_{n(c)}) = 1 - \epsilon$ , and  $f_W^{(c)}(x_{n(c')}) = \frac{\epsilon}{C-1}$ . Under the conditions of NC1 we know that all feature vectors in a

<sup>18</sup>In the binary case the last layer weights are  $V_L^{c=+1}$  and  $V_L^{c=-1}$ .

class collapse to the class mean, ie  $h(x_{n(c)}) = \mu_c$ . Let us denote the global feature mean by  $\mu_G = \frac{1}{C} \sum_c \mu_c$ . This means we have:

$$\begin{aligned} \dot{W}_L^c &= \epsilon \times N \times \mu_c - \frac{\epsilon}{C-1} \times N \times \sum_{c' \neq c} \mu_{c'} - \lambda W_L^c = 0 \\ \implies W_L^c &= \frac{CN\epsilon}{\lambda(C-1)} \times (\mu_c - \mu_G) \end{aligned} \quad (54)$$

This means that the last layer parameters  $W_L$  are a scaled version of the centered class-wise feature matrix  $M = [\dots \mu_c - \mu_G \dots]$ . This means that at equilibrium, with quasi interpolation of the training labels, we have  $\frac{W_L}{\|W_L\|_F} = \frac{M^\top}{\|M\|_F}$ . 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 ( $W_L^c$ , and hence  $\mu_c - \mu_G$ ) have the same norm:

$$\begin{aligned} \dot{W}_L^c &= \sum_n (y_n^c - \langle W_L^c, h(x_n) \rangle) h(x_n) - \lambda W_L^c = 0 \\ \implies \langle W_L^c, \dot{W}_L^c \rangle &= \sum_n (y_n^c - f_W^{(c)}(x_n)) f_W^{(c)}(x_n) - \lambda \|W_L^c\|_2^2 = 0 \\ \implies \|W_L^c\|_2^2 &= \frac{N}{\lambda} \left( \epsilon - \frac{C}{C-1} \epsilon^2 \right) \end{aligned} \quad (55)$$

From the quasi-interpolation of the correct class we have that

$$\begin{aligned} \langle W_L^c, \mu_c \rangle &= 1 - \epsilon \\ \implies \langle W_L^c, \mu_G \rangle + \langle W_L^c, \mu_c - \mu_G \rangle &= 1 - \epsilon \\ \implies \langle W_L^c, \mu_G \rangle &= 1 - \epsilon - \frac{\lambda(C-1)}{CN\epsilon} \|W_L^c\|^2 \\ &= 1 - \epsilon - \frac{\lambda(C-1)}{CN\epsilon} \times \frac{N}{\lambda} \left( \epsilon - \frac{C}{C-1} \epsilon^2 \right) \\ \implies \langle W_L^c, \mu_G \rangle &= \frac{1}{C} \end{aligned} \quad (56)$$

From the quasi-interpolation of the incorrect classes, we have that

$$\begin{aligned}
& \langle W_L^c, \mu_{c'} \rangle = \frac{\epsilon}{C-1} \\
\implies & \langle W_L^c, \mu_{c'} - \mu_G \rangle + \langle W_L^c, \mu_G \rangle = \frac{\epsilon}{C-1} \\
& \implies \frac{\lambda(C-1)}{CN\epsilon} \times \langle W_L^c, W_L^{c'} \rangle = \frac{\epsilon}{C-1} - \frac{1}{C} \\
\implies & \|W_L^c\|_2^2 \times \frac{\lambda(C-1)}{CN\epsilon} \times \langle V_L^c, V_L^{c'} \rangle = \frac{\epsilon}{C-1} - \frac{1}{C} \\
& \implies \langle V_L^c, V_L^{c'} \rangle = -\frac{1}{C-1}
\end{aligned} \tag{57}$$

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 have that  $\sum_c W_L^c = 0$ . This completes the proof of the NC2 condition.

We note that for the square loss these results apply to each global minimum (that is, close-to-zero square-loss), irrespectively of its  $\rho_{eq}$  and, therefore, irrespectively of how good the associated expected error is.

## G.2 Exponential loss

**Corollary 5** *Under the assumptions of square loss with Batch Normalization and Weight Decay, the SGD equations imply NC1, NC2, NC3, NC4.*

For the exponential loss with normalization and weight decay the gradient flow corresponding to GD are  $\dot{\rho}_k = \frac{1}{N} \sum_n e^{-\rho y_n f_n} y_n f_n - \lambda \rho$  and  $\dot{V}_k = \rho \frac{1}{N} \sum_n e^{-\rho y_n f_n} y_n S_k \frac{\partial f_n}{\partial V_k}$

For SGD with minibatch size = 1 the equations at equilibrium are

$$\begin{aligned}
0 &= \frac{1}{N} \sum_n e^{-\rho y_n f_n} y_n f_n - \lambda \rho \\
0 &= \rho \frac{1}{N} \sum_n e^{-\rho y_n f_n} y_n S_k \frac{\partial f_n}{\partial V_k}
\end{aligned} \tag{58}$$

for  $k = 1, \dots, L-1$  is (with  $S_k = I - V_k V_k^T$ ).

For GD a critical point with  $\dot{\rho} = 0$  implies  $\lambda \rho = \frac{1}{N} \sum_n e^{-\rho y_n f_n} y_n f_n$  and thus at  $\dot{V}_k = 0$  it implies

$$\sum_n e^{-\rho y_n f_n} y_n \frac{\partial f_n}{\partial V_k} = V_k \sum_n e^{-\rho y_n f_n} y_n f_n = \lambda \rho V_k \tag{59}$$

The condition above does not by itself imply that all the margins  $f_n$  are the same (which is required for NC1). The situation is however different for SGD: equilibrium for SGD with

minibatch size of 1 (the argument is valid also for minibatch sizes larger than 1 but smaller than  $N[10]$ ) implies

$$e^{-\rho y_n f_n} y_n f_n = \lambda \rho, \quad \forall n = 1, \dots, N \quad (60)$$

and thus  $f_1 = f_n, \quad \forall n = 1, \dots, N$ .

With result the same arguments used for the square loss can be used to prove NC1 to NC4 with small changes as show here.

- NC1: as before since the margins  $y_i f_i$  are equal for all  $i$ . Then  $V_L h_{i,c} = f^c$  is also independent of  $i$  implying that  $h_{i,c}$  is independent of  $i$  at convergence.
- NC2: see above.
- NC3: see above.
- NC4: theorem 2 of [16] states that in the exponential loss case, NC1 and NC2 imply NC3 and NC4.

The proof above for NC1 requires  $\lambda > 0$ . In the square loss case it is also required but for a superficially different reason (to ensure convergence of the dynamical system).

The proof for the exponential loss requires SGD, unlike the square loss case. We do not know whether this 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.*

## H Bias towards low-rank weight matrices

An intriguing argument for small rank weight matrices is the following observation:

**Lemma 6** *At SGD convergence – that is when  $\dot{\rho} = 0, \quad \dot{V}_k = 0, \quad \forall S$  where  $S$  is a minibatch – for  $\lambda > 0$  the matrices  $V_k$  have rank 1.*

The lemma follows assuming that  $\dot{V}_k = 0$  for all minibatches at convergence for  $\lambda > 0$ . This implies

$$V_k f_n = \frac{\partial f_n}{\partial V_k}, \quad \forall k = 1, \dots, L, \quad \forall n \quad (61)$$

To see why suppose

$$f(x) = (V_L \sigma(V_{L-1} \dots \sigma(V_1 x))) \quad (62)$$

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

$$f(x_j) = V_L D_{L-1}(x_j) V_{L-1} \cdots V_{k+1} D_k(x_j) V_k \cdots D_1(x_j) V_1 x_j \quad (63)$$

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  $D_{k-1}(x_j) = \text{diag}[\sigma'(N_k(x_j))]$  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  $\frac{\partial f}{\partial V_k} = ab^T$ <sup>19</sup>. As sanity checks,  $f^T = b^T V_k^T a = f$ ; furthermore, the structural lemma Equation 1 gives

$$\sum_{i,j} \frac{\partial f(V; x)}{\partial V_k^{i,j}} V_k^{i,j} = \sum_{i,j} a^i b^j V_k^{i,j} = f(x). \quad (64)$$

Then Equation 61 becomes

$$V_k f = [V_L D_{L-1}(x) V_{L-1} \cdots V_{k+1} D_k(x)]^T D_{k-1}(x) V_{k-1} D_{k-2}(x) \cdots D_1(x) V_1 x \quad (65)$$

Then Equation 61 becomes

$$V_k f = [V_L D_{L-1}(x) V_{L-1} \cdots V_{k+1} D_k(x)]^T D_{k-1}(x) V_{k-1} D_{k-2}(x) \cdots D_1(x) V_1 x = ab^T \quad (66)$$

Thus  $V_k$  is a rank one matrix. This means that for exampl[e when  $L = 2$  that

$$f(x) = a_2 D_1(x) a_1 b_1^T x \quad (67)$$

### Remarks

- Notice that if each weight matrix is of rank one or close to rank one, the best way to add parameters to the network of a given width is to increase its depth: the number of effective parameters will increase linearly with depth.
- 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.
- BN should correspond to normalizing each row of each weight matrix, thereby effectively normalizing each weight matrix. Thus the argument developed here for LM should apply to networks normalized with BN though the dynamics will be different.

An intuition of 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  $\rho$ . This suggests minimization of the "stable rank", defined as the ratio of the square of its Frobenius norm and the square of its spectral norm  $\frac{\|V\|_F^2}{\|V\|^2} = \frac{\sum_j \sigma_j^2}{\max_i \sigma_i^2}$ .

<sup>19</sup>When  $f = a^T b$ , then  $\frac{\partial f}{\partial a} = b$  and  $\frac{\partial f}{\partial b} = a$

# I Blue-sky Remarks on Orthogonality

Let us now make some strong assumptions to get some intuition about the potential impact of the constraints on the weight matrices.

*Assumptions*

- Let us assume that all  $V_k \quad \forall k = 1, \dots, (L - 1) \in \mathbb{R}^{p,p}$  have the same dimensions, whereas  $V_L \in \mathbb{R}^{1,p}$ .
- We assume linear deep networks *at training time*, that is without RELUs. This is inspired by the observation that *if solutions  $V_k$  are found that satisfy Equation 65 then they will also satisfy the same Equations when the matrices  $D_k$  are replaced by  $I$ , while the converse is not true.*

The intuition is that this big oversimplification may still be interesting, because the “training” equations above have to hold for all the  $x_n$  in the training set. This implies that the  $D$  matrices at each level are likely to eventually have 1 in each position of the diagonal across the whole of the training set. Of course, this will not hold completely and for all layers, especially if  $p > N$  and especially for the layers at the top of the network, in which case the presence of the  $D_k$  makes the constraint Equation 61 effectively weaker.

Under these three assumptions, let us consider two natural types of solutions that are consistent with Equation 61.

## I.1 $V_k$ as projection matrices

A class of solution which is consistent with the constraints represented by Equation 61 is

$$V_1 = V_2 = \dots = V_{L-1} \tag{68}$$

and

$$V_L = D_{L-1}(x)V_{L-1} \dots \dots V_{k+1}D_k(x)V_k \dots D_1(x)V_1x = \frac{\partial f}{\partial V_L} \tag{69}$$

In order for this to be true, the  $V_k \quad k < L$  matrices can be projection matrices ( $P$  is a projection if  $P^2 = P$ ; it is an orthogonal projection if  $P = P^T$ ). Then, all the weight matrices are proportional to each other apart from the weight matrix of the last ( $L$ ) layer which must be a vector proportional to the vector of activities of the units in layer  $L - 1$ .

If we assume that feedforward networks with  $T$  layers converge to this type of solution, the interesting *prediction is that recurrent networks (therefore with weight sharing across layers) under  $T$  iterations should be identical to forward networks with  $T$  layers (without weight sharing).*

## I.2 $V_k$ as orthogonal matrices

Another possible set of solutions consists of matrices  $V_k$  (assuming the weight matrices are all square matrices) each proportional to an orthogonal matrix. The constraint Equations 61 suggest the structure of a group since  $V_k$  is proportional to the product of similar matrices.

A key property of orthogonal matrices is that  $V_k^T = V_k^{-1}$ . Because of this property the constraint equations are always satisfied. For instance assume  $f(x) = V_4 V_3 V_2 V_1 x$  with the matrices being orthogonal. Then it is easy to check that the constraint equations yield  $V_3 \propto \frac{\partial f}{\partial V_3} = V_4^T (V_2 V_1)^T$  and  $V_2 \propto (V_4 V_3)^T (V_1)^T$ . Together they satisfy  $V_3 = V_3$ .

We observe that the underlying reason for restricting this class of solutions to the orthogonal group is BN or WN, since they are equivalent to constrained optimization with Lagrange multipliers. As observed in [23] regularization of each weight matrix of a linear network reduces the symmetry group of the loss function from  $GL_p(\mathbb{R})$  to the orthogonal group  $O_p(\mathbb{R})$ . Furthermore, it is interesting to notice, as they do, that

- Orthogonal matrices are the determinant  $\pm 1$  matrices of minimal Frobenius norm (the squared determinant is the product of the squared singular values);
- Orthogonal matrices are the inverse matrices of minimum total squared Frobenius norm (sum of the squared singular values);
- A square matrix is orthogonal iff  $A^{-1} = A^T$ ;
- Orthogonal matrices diagonalize any symmetric real-valued matrix  $A = U \Lambda U^T$ .

There is a large number of papers (for a random one see [24] and references therein) discussing the advantages of orthogonality for generalization in deep networks and probably as many papers proposing regularization-like algorithms in order to impose orthogonality in the weight matrices in a deep network. More generally, the discussion above should be extended from orthogonal matrices to non-square matrices in an orthogonal Stiefel manifold on the sphere. As far as we know, this appendix represents the first time that commonly used normalization algorithms, such as BN, are shown to bias weight matrices towards being orthogonal (or projection) vectors. It is natural to conjecture that additional properties of deep networks may be derived from the rich structure induced by this bias. On the other hand, we cannot expect weight matrices to be orthogonal in real networks because of the role of the RELUs, which is not taken into account in the simplified analysis above, and because BN in practice does not exactly normalize the weight matrices<sup>20</sup>.

## I.3 Diagonal networks (with A. Banburski)

Diagonal networks have been recently analyzed in a number of theoretical papers (see [19] and references therein) and are particularly interesting here for two reasons which we show below:

---

<sup>20</sup>Normalization of each row exactly to norm 1 implies normalization of the matrix to  $M$  where  $M$  is the number of rows. However, normalization of each row to different constants does not lead to orthogonality.



diagonal initializations are preserved by gradient descent and the D-matrices corresponding to the RELUs stages commute.

First, notice that gradient updates are given by

$$w_{t+1} - wt = -\eta \sum_n \ell(f(w; x_n), y_n)' \nabla_{w_t} f(w; x_n) \quad (70)$$

and so an update to any off-diagonal weight will be non-zero only if  $\nabla_{w_t} f(w; x_n)$  for that weight is non-zero. We trivially find that

$$\frac{\partial f(x)}{\partial W_k^{ij}} = W_L D_{L-1}(x) W_{L-1} \cdots W_{k+1}^i D_k(x) D_{k-1}(x) W_{k-1}^j D_{k-2}(x) \cdots D_1(x) W_1 x. \quad (71)$$

This means that a off-diagonal layer- $k$  entry  $(i, j)$  update depends on a term in  $f$  that has an  $i$ -th column in layer  $k + 1$  and  $j$ -th row in layer  $k - 1$ . But if at time  $t = 0$  the off-diagonal terms  $i \neq j$  vanish, then the off-diagonal gradient update also vanishes for  $t = 1$  and similarly for all subsequent times. Hence GD preserves diagonal initial conditions.

Assuming the same setting of square matrices as in the previous subsections, we can immediately see the usefulness of the diagonal assumption, from the simple fact that diagonal square matrices commute, so we can write

$$f = V_L \dots V_1 D_{L-1}(x) \dots D_1(x) x = V_L \dots V_1 \tilde{x} \quad \text{with} \quad \tilde{x} = D_{L-1}(x) \dots D_1(x) x,$$

i.e. we can push all the nonlinearities to the end and absorb them now into a single nonlinear transformation of the data  $x$ . This then allows us to deal with the sum over all training examples: by defining  $\hat{x} = \sum_n \alpha_n \tilde{x}_n$  we have  $\hat{f} = V_L \dots V_1 \hat{x}$ .

We can now write the equation at the critical point here as

$$\frac{\partial \hat{f}}{\partial V_k^i} = V_L^i \cdots V_{k+1}^i V_{k-1}^i \cdots \hat{x}^i$$

and

$$V_k^i \hat{f} = V_k^i V_L \cdots V_{k+1} V_k V_{k-1} \cdots \hat{x}$$

where we now can label the diagonal layers with a single index  $i$ . Let us look at a specific example of a 4-layer network now,  $\hat{f} = V_4 V_3 V_2 V_1 \hat{x}$ . The equations are of the form

$$V_4^i \hat{f} = V_3^i V_2^i V_1^i \hat{x}^i$$

and similarly for other layers. Solving these, we get the very interesting constraints  $\hat{f}^2 = (V_2^i V_1^i \hat{x}^i)^2 = (V_3^i V_1^i \hat{x}^i)^2 = \dots$ , or that some  $V_k^i = 0$ . This gives us that at the critical point,  $V_k^i = \pm \frac{\hat{x}^i}{\hat{f}}$  or  $V_k^i = 0$ .

Putting back the definitions for the different expressions, we get that at the critical points, the normalized weights are given by

$$V_k^i = \pm \frac{\sum_n \alpha_n D_{L-1}(x_n^i) \dots D_1(x_n^i) x_n^i}{\sum_n \alpha_n f(V; x_n)}, \quad (72)$$

with  $i = 1, \dots, p$  being one of the input channels (and the diagonal path through the network). Notice that the dependence on the layer  $k$  is only in the  $\pm$  sign.