Generalization Puzzles in Deep Networks

Anonymous authors
Paper under double-blind review

Abstract

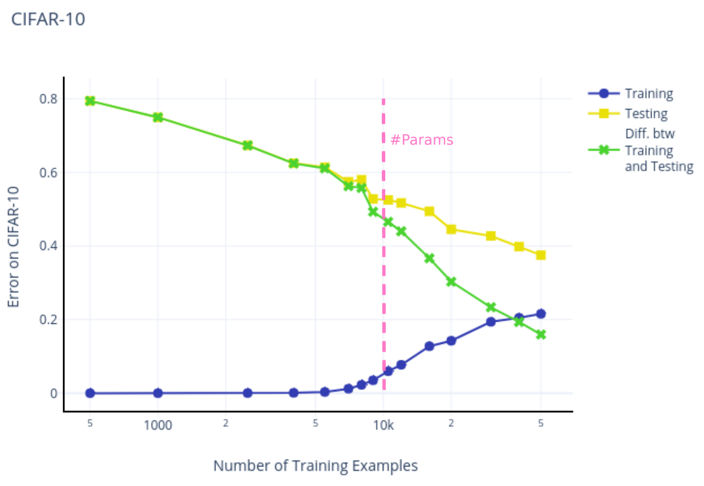
In the last few years, deep learning has been tremendously successful in many applications. However, our theoretical understanding of deep learning, and thus the ability of providing principled improvements, seems to lag behind. A theoretical puzzle concerns the ability of deep networks to predict well despite overparametrization and despite the lack of complexity control under the form of explicit regularization. Do deep networks require a drastically new theory of generalization? In particular, are there measurements based on the training data that are predictive of the network performance on future data? Here we show that when performance is measured appropriately, the training performance is in fact predictive of expected performance, consistently with classical machine learning theory and the presence of an implicit regularization.

1 Introduction

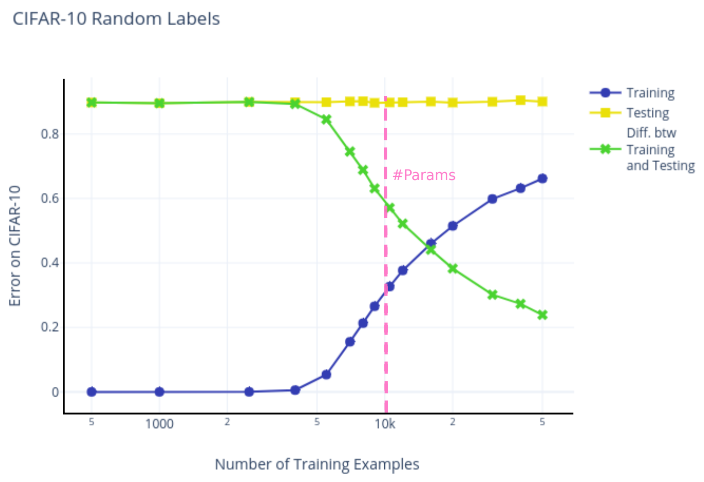
Is it possible to decide the prediction performance of a deep network from its performance in training – as it is typically the case for shallower classifiers such as kernel machines and linear classifiers? Is there any relationship at all between training and test performances? Figure 1a shows that when the network has more parameters than the size of the training set – which is the standard regime for deep nets – the training classification error can be zero and is very different from the testing error. This intriguing lack of generalization was recently highlighted by the surprising and influential observation (Zhang et al. (2016)) that the same network that predicts well on normally labeled data (CIFAR10), can fit randomly labeled images with zero classification error in training while its test classification error is of course at chance level, see Figure 1b. The riddle of large capacity and good predictive performance led to many papers, with a variety of claims ranging from "This situation poses a conceptual challenge to statistical learning theory as traditional measures of model complexity struggle to explain the generalization ability of large artificial neural networks..." Zhang et al. (2016), to various hypotheses about the role of flat minima Keskar et al. (2016); Dinh et al. (2017); Chaudhari et al. (2016), about SGD Chaudhari & Soatto (2017); Zhang et al. (2017) and to a number of other explanations (e.g. Belkin et al. (2018); Martin & Mahoney (2019)) for such unusual properties of deep networks.

We start by defining some key concepts. We call "loss" the measure of performance of the network f on a training set $S = x_1, y_1, \cdots, x_N, y_N$. The most common loss optimized during training for binary classification is the logistic loss $L(f) = \frac{1}{N}\sum_{n=1}^{N} \ln(1 + e^{-y_n f(x_n)})$. We call classification "error" $\frac{1}{N}\sum_{n=1}^{N} H(-y_n f(x_n))$, where y is binary and H is the Heaviside function with $H(-yf(x)) = 1$ if $-yf > 0$ which correspond to wrong classification. There is a close relation between the logistic loss and the classification error: the logistic loss is an upper bound for the classification error. Thus minimizing the logistic loss implies minimizing the classification error.

The criticism in papers such as Zhang et al. (2016) refers to the classification error. However, training minimizes the logistic loss. As a first step it seems therefore natural to look at whether logistic loss in training can be used as a proxy for the logistic loss at testing. The second step follows from the following observation. The logistic loss can always be made arbitrarily small for separable data (when $f(x_n)y_n > 0, \forall n$) by scaling up the value of f and in fact it can be shown that the norm of the weights of f grows monotonically with time



(a)



(b)

Figure 1: *(a) Generalization error (yellow, the difference between training and test classification accuracy) in CIFAR10 and (b) generalization error in CIFAR10 with random labels. The DNN, trained by minimizing the cross-entropy loss, is a 5-layer convolutional network (i.e., no pooling) with 16 channels per hidden layer. The activation are ReLUs. The resulting architecture has approximately* 10000 *parameters. SGD was used with batch size* = 100 *for* 70 *epochs for each point. Neither data augmentation nor regularization is performed. The generalization error in classification seems to converge to zero for increasing size of the training set. In these plots we use up to all the training examples available for CIFAR-10 (50K).*

during gradient descent. Notice that multiplying f by any positive number does not affect its sign and thus does not change its behavior at classification. The second step is then to consider the logistic loss of the *normalized network* $\tilde{f}$ with $f = \rho\tilde{f}$, with $\rho = \rho_1...\rho_K$ where ρ_i is the Frobenius norm of the weight matrix of layer i.

Now we show that by measuring the performance of a trained network on the training set in terms of the quantities described above, classical generalization holds: the performance of the network on the training set becomes a good qualitative proxy for the expected future performance of the network. The precise procedure involves normalizing the output of the trained network by dividing it by a single number – the product of the Frobenius norms of the weight matrices at each layer – and then measuring the cross-entropy loss.

Figure 2a) shows the cross entropy test loss plotted versus the training loss for networks with the same architecture trained with different initializations (this is a way to get networks with different test performance). There is no clear relation between training and test cross-entropy loss in the same way that there is not between classification error in training and testing. Normalizing each network separately yields Figure 2b): now the test loss is tightly predicted by the training loss. We emphasize that normalization does not change the classification performance of the networks. In fact, since (binary) classification depends only on the sign of $f(x)$, *it is not changed by normalization to any ball.* The figure suggests that the empirical cross-entropy loss of $\tilde{f}$ on the training set predicts rather well how networks will perform on a test set. Impressively, the same normalization process correctly predicts the test loss when the training is on randomly labeled data. This apparent lack of predictivity of the training performance on randomly labeled data was at the core of Zhang et al. criticism of machine learning theory (Zhang et al. (2016)). Figure 2 shows that it is in fact possible to gauge just from the training performance that the network trained on randomly labeled examples will have chance performance on a test set even if its *classification error* in training is always zero. As shown in Figure 2 the data point corresponding to the randomly trained network still satisfies approximately the same linear relationship, as explained by the classical theory. Thus, while Figure 2 shows that the normalized train cross-entropy can predict the performance of the of the normalized test cross-entropy, Figure 4 b shows that the normalized test cross-entropy is approximately predictive of the relative test classification error. Thus Figures 2 and 4 b together show that the normalized train cross-entropy loss in training is approximately predictive of the test classification error in testing, at least in terms of ranking between networks.

2 Interpretation in terms of classical learning theory

We define a deep network with K layers with the usual elementwise scalar activation functions $\sigma(z): \quad \mathbf{R} \to \mathbf{R}$ as the set of functions $f(W_1, \cdots, W_K; x) = \sigma(W_K\sigma(W_{K-1} \cdots \sigma(W_1 x)))$, 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 use the symbol W as a shorthand for the set of W_k matrices $k = 1, \cdots, K$. For simplicity we consider here the case of binary classification in which f takes scalar values, implying that the last layer matrix W_K is $W_K \in \mathbf{R}^{1,K_l}$ and the labels are $y_n \in \{-1, 1\}$. There are no biases apart form the input layer where the bias is instantiated by one of the input dimensions being a constant. The activation function in this paper is the ReLU activation. We denote the network as $f = f(W_1, \cdots, W_K; x)$ where x is the input and the weight matrices W_k are the parameters. We call $\tilde{f}$ the network with normalized weights matrices, that is $\tilde{f} = f(\tilde{V_1}, \cdots, \tilde{V_K}; x)$ with $V_i = \frac{W_i}{||W_i||}$.

Consider different asymptotic minima of the empirical loss obtained with the same network architecture on the same training set by minimization of the cross-entropy loss with different initial conditions (see Appendix). We obtain different test losses, depending on initial conditions. The question is whether their test performance can be predicted from empirical properties measured only on the training set.

CIFAR10, MNIST and CIFAR100 [1] used in our experiments are multiclass problems. In this theory section we discuss for simplicity the simpler case of binary classification. Exactly the same steps in the theoretical arguments below apply to the cross-entropy loss (because Equations 1,3 apply, see Appendix).

Consider the structure of the loss for a deep network. The "positive homogeneity" property of ReLU networks implies the following property:

$$f(W_1, \cdots, W_K; x) = \rho_1 \cdots \rho_K \tilde{f}(x) = \rho \tilde{f}(x) \tag{1}$$

where $W_K = \rho_k \tilde{W_K}$ and $||\tilde{W_K}|| = 1$ and $\rho_1 \cdots \rho_K = \rho$.

[1]MNIST and CIFAR100 results are in the appendix

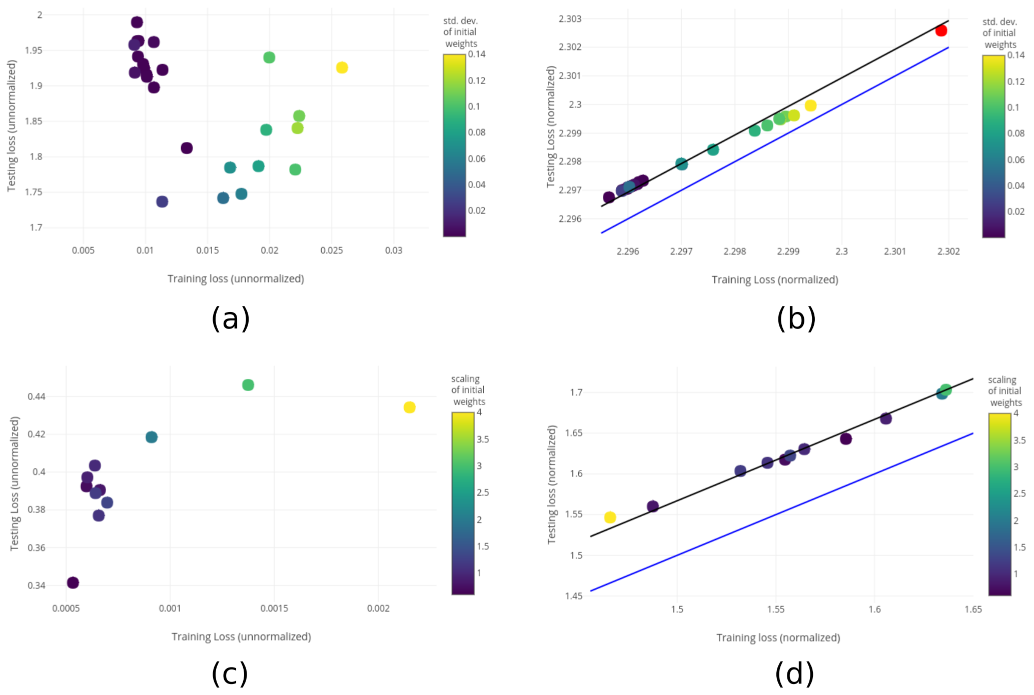


Figure 2: *The plot in a) shows testing vs training cross-entropy loss for networks trained on the same data sets but with different initializations. The graph in b) shows the testing vs training loss for the same networks, normalized by dividing each weight by the Frobenius norm of its layer. Notice that all points have zero classification error at training. The red point in b) refers to a network trained on the same CIFAR-10 data set but with randomized labels. It shows zero classification error at training and test error at chance level. The black line is a square-loss regression of slope* 1 *with positive intercept. The blue line is the diagonal at which training and test loss are equal. The networks are 3-layer convolutional networks. The plots in c) and d) show similar experiments for ResNet-56, demonstrating that our observations hold for state-of-the-art networks (testing errors ranging from 7% to 9%; on this 10-classes classification chance performance is* 90% *error). We again emphasize that the performance in classification of normalized and unnormalized networks is exactly the same. The normalization in the case of ResNet was performed by using the norm of the output of each network on one example from CIFAR-10 (randomly chosen), because we found it computationally difficult because of skip connections to directly evaluate the effective norm of the residual layers. Further details of this experiment are in the Appendix.*

This property is valid for layer-wise normalization under any norm. We emphasize again that $f(W_1, \cdots, W_K; x)$ and $\tilde{f}(x) := f(\tilde{W}_1, \cdots, \tilde{W}_K; x)$ have the same classification performance on any data set. Furthermore, during gradient descent on separable data – most data sets are separable by overparametrized deep networks – it can be shown (see Poggio et al. (2018)) that the ρ factor continue to increase with time towards infinity, driving an exponential type loss to zero without affecting the classification performance, which only depends on the sign of $y_n f(x_n) \forall n$. As we discussed earlier, it seems that different networks corresponding to different empirical minimizers[2] could be evaluated in terms of their normalized form $\tilde{f}(x)$. The intuition is similar to the linear case in which the classifier $w^T x$ depends on the unit vector $\tilde{w} = \frac{w}{|w|}$ while $|w|$ diverges to infinity Soudry et al. (2017); Poggio et al. (2018).

[2]In general an overparametrized network may have a large number of global minima, see for instance Poggio & Liao (2017).

To assess $\tilde{f}(x)$ we compute its logistic loss (which is the special case of cross-entropy in the binary case)

$$\hat{L}(\tilde{f}) = \frac{1}{N}\sum_{n=1}^{N} \ln(1 + e^{-y_n \tilde{f}(x_n)}). \tag{2}$$

Of course for separable data $(y_n f(x_n) > 0, \forall n)$ the loss of $\tilde{f}$ is larger than the loss of f since the negative exponent is smaller. Figure 2 uses L_2 for layer-wise normalization.

We are now ready to explain the results of Figure 2 in terms of classical machine learning theory. A typical generalization bound that holds with probability at least $(1-\delta)$, $\forall f \in \mathbb{F}$ has the form Bousquet et al. (2003):

$$|L(f) - \hat{L}(f)| \leq c_1 \mathbb{R}_N(\mathbb{F}) + c_2 \sqrt{\frac{\ln(\frac{1}{\delta})}{2N}} \tag{3}$$

where $L(f) = \mathbf{E}[\ell(f(x), y)]$ is the expected loss, $\mathbb{R}_N(\mathbb{F})$ is the empirical Rademacher average of the class of functions $\mathbb{F}$ measuring its complexity; c_1, c_2 are constants that reflect the Lipschitz constant of the loss function and the architecture of the network.

We use the bound in Equation 3 and the key observation that the Rademacher complexity satisfies the property,

$$\mathbb{R}_N(\mathbb{F}) = \rho \mathbb{R}_N(\tilde{\mathbb{F}}), \tag{4}$$

because of homogeneity of the networks. Then, the bound on the cross-entropy loss for the unnormalized network gives

$$L(f) \leq \prod_{k=1}^{K} \mathbb{R}_N(\mathbb{F}) + c_2 \sqrt{\frac{\ln(\frac{1}{\delta})}{2N}}, \tag{5}$$

since $\hat{L}(f) \approx 0$. Considering the corresponding bound for the cross-entropy loss of the normalized network *scaled with any desired scaling R* gives

$$L(R\tilde{f}) \leq \hat{L}(R\tilde{f}) + R\mathbb{R}_N(\tilde{\mathbb{F}}) + c_2 \sqrt{\frac{\ln(\frac{1}{\delta})}{2N}}. \tag{6}$$

In our experiments we find that $\mathbb{R}_N(\tilde{\mathbb{F}})$ is small for the value of N of our datasets and $\mathbb{R}_N(\mathbb{F})$ is large. Equation 5 implies then that the unnormalized test loss will be quite different from zero and thus different from the unnormalized train loss. On the other hand, in Equation 6 the terms $\mathbb{R}_N(\tilde{\mathbb{F}}) + c_2\sqrt{\frac{\ln(\frac{1}{\delta})}{2N}}$ are small, implying that the normalized test loss is very close to the normalized train loss. Thus Equation 6 with $R = 1$ shows that $L(\tilde{f})$ is bounded by $\hat{L}(\tilde{f})$, predicting a bound in terms of a linear relationship with slope one and a small offset between $L(\tilde{f})$ and $\hat{L}(\tilde{f})$. The prediction is verified experimentally (see Figure 2). Notice that both homogeneity of the ReLU network and separability, that is very small cross-entropy loss, are key assumptions in our argument. The first applies to networks with a linear or ReLU activation but not to networks with other activation functions. The second is usually satisfied by overparametrized networks. Thus Equation 6 shows that the training cross-entropy loss is a very good proxy of the cross-entropy loss at testing, implying generalization for a relatively small number of examples N . Notice that for all the networks in Figure 2, classification performance at training is perfect and that scaling does not change the sign of the networks outputs and therefore their behaviour in terms of classification. In particular, Figure 2 shows that performance on the randomly labeled training set when measured for the normalized network is bad (despite classification being at zero error) predicting correctly bad performance on the test set. As we mentioned, Figure 4 b shows that there is a good correlation between crossentropy loss and classification performance: empirically the ranking between the different classifiers is mostly the same for crossentropy vs classification loss.

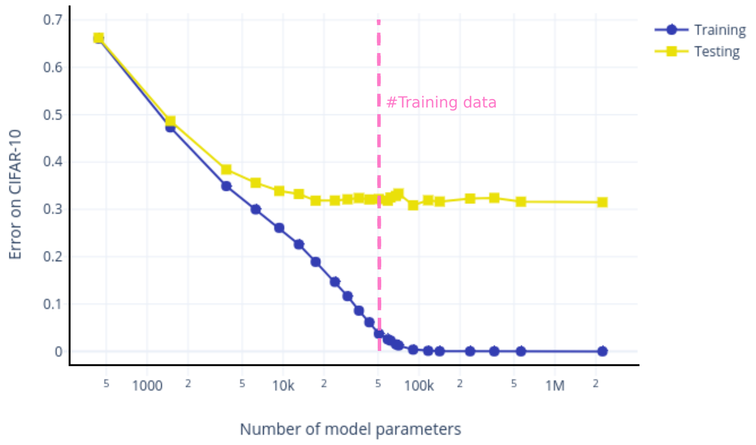


Figure 3: *Classification error in CIFAR-10 as a function of number of parameters for fixed training set. The DNN is the same as in Figure 1. The classification error at test does not increase here when increasing the number of parameters beyond the size of the training set .*

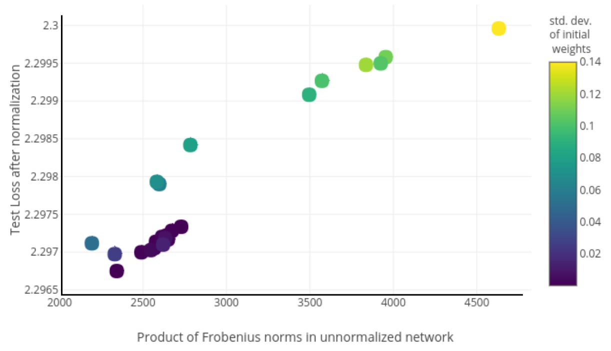
3 Minimization of the cross-entropy loss implies minimization of the classification error

Gradient descent techniques, such as stochastic gradient descent (SGD), are used to minimize the cross-entropy loss in deep networks. This is a typical approach in machine learning: minimize a convex *surrogate* of the $0 - 1$ loss function used for binary classification. The logistic loss is an upper bound to the binary classification error: thus minimization of the loss implies minimization of the error, yielding a monotonic relation between the logistic loss and the classification error (Figure 4 b). Thus $\hat{L}(f)$ is a much better bound on the value of the classification error than $\hat{L}(\tilde{f})$. Figure 4 c shows the test classification error $R(f)$ as a function of the product of the layerwise norms of the network. Notice that on this 10-classes classification problem chance performance is 90% error. Its monotonic behavior, consistent with previous reports Neyshabur et al. (2017); Liang et al. (2017), follows from Equation 5 and the fact that $R(f)$ is upper-bounded by the expected cross-entropy loss $L(f)$.

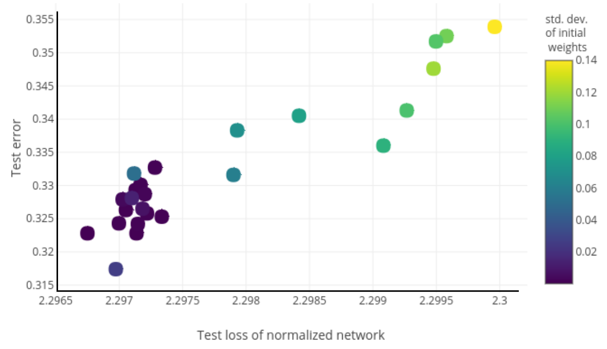
The linear behavior of a bound on the expected loss as a function of the empirical loss $L(\tilde{f})$ is thus expected and it is a consequence of previous results Neyshabur et al. (2017); Liang et al. (2017); Bartlett et al. (2017); Neyshabur et al. (2017) and of the upper bounds Equations 3 and 6. Lower bounds, however, are not available. As a consequence, the theory does not guarantee that among two (normalized) networks, the one with lower cross-entropy loss in training will always have a lower classification error at test. This difficulty is not specific to deep networks. It is common to approaches using a surrogate loss function. The empirical evidence however supports the claim that there is a roughly monotonic relationship between training (and testing) loss of the normalized network and its expected classification error: Figure 4b shows an approximately monotonic relation between normalized test cross-entropy loss and test classification error.

4 Discussion

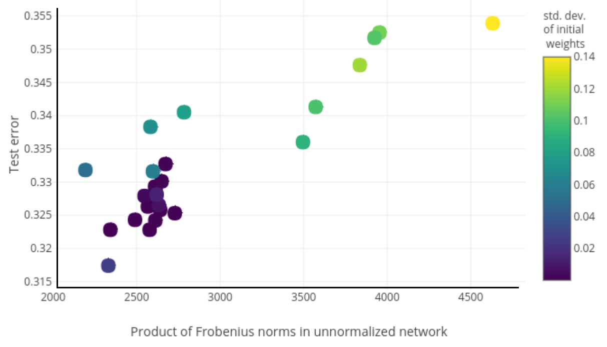
The linear relationship we found means that the generalization error of Equation 3 is *small* once the complexity of the space of deep networks is "dialed-down" by normalization. It also means that, as expected from the theory of uniform convergence, the generalization gap decreases to zero for increasing size of the training set (see Figure 1). Thus there is



(a)



(b)



(c)

Figure 4: *a) Test loss for the normalized networks vs the product of the unnormalized Frobenius norms of the layers, b) test classification error as a function of the normalized test loss and c) test classification error vs the product of the unnormalized Frobenius norms of the layers. The network architecture is the same as in the top of Figure 2.*

indeed asymptotic generalization - defined as training loss converging to test loss when the number of training examples grows to infinity - in deep neural networks, when appropriately measured.

The title in Zhang et al. (2016) "Understanding deep learning requires rethinking generalization" seems to suggest that deep networks are so "magical" to be beyond the reach of existing machine learning theory. This paper shows that *this is not the case*. On the other hand, the generalization gap for the classification error and for the unnormalized cross-entropy is expected to be small only for much larger N (N must be significantly larger than the number of parameters). However, consistently with classical learning theory, the cross-entropy loss at training predicts well the cross-entropy loss at test when the complexity of the function space is reduced by appropriate normalization. For the normalized case with $R = 1$ this happens in our data sets for a relatively "small" number N of training examples as shown by the linear relationship of Figure 2.

The classical analysis of ERM algorithms studies their asymptotic behavior for the number of data N going to infinity. In this limiting regime, $N > W$ where W is the fixed number of weights; consistency (informally the expected error of the empirical minimizer converges to the best in the class) and generalization (the empirical error of the minimizer converges to the expected error of the minimizer) are equivalent. This note implies that there is indeed asymptotic generalization and consistency in deep networks. However, it has been shown that in the case of linear regression, for instance with kernels, there are situations – depending on the kernel and the data – in which there is simultaneously interpolation of the training data and good expected error. This is typically when $W > N$ and corresponds to the limit for $\lambda = 0$ of regularization, that is the pseudoinverse. It is likely that deep nets may have a similar regime, in which case the implicit regularization described here, with its asymptotic generalization effect, is just an important prerequisite for a full explanation for $W > N$ – as it is the case for kernel machines under the square loss.

The results of this paper strongly suggested that the complexity of the normalized network is controlled by the optimization process. In fact a satisfactory theory of the precise underlying implicit regularization mechanism has now been proposed Soudry et al. (2017); Du et al. (2018); Banburski et al. (2019); Shpigel Nacson et al. (2018).

As expected, the linear relationship we found holds in a robust way for networks with different architectures, different data sets and different initializations.

Our observations, which are mostly relevant for theory, yield a recommendation for practitioners: it is better to monitor during training the empirical "normalized" cross-entropy loss instead of the unnormalized cross-entropy loss actually minimized. The former matters in terms of stopping time and predicts test performance in terms of cross-entropy and ranking of classification error. More significantly for the theory of Deep Learning, this paper confirms that classical machine learning theory can describe how training performance is a proxy for testing performance of deep networks.

A Multi-class classification extension

While the theoretical explanation in the main text applies to the case of binary classification, the extension to multi-class case follows straightforwardly.

Recall some definitions for neural networks with multiple outputs. Let C be the number of classes – the neural network is then a vector $f(W;x) \in \mathbf{R}^C$. The component $f_j(W;x)$ denotes the j-th output. The dataset is again composed of examples $x_n \in \mathbf{R}^d$ and labels are now $y_n \in [C]$. Note that nothing here changes in regards to homogeneity, and we again can define a normalized network

$$f(W_1, \cdots, W_K; x) = \rho_1 \cdots \rho_K \tilde{f}(x) \tag{7}$$

The main theoretical arguments depend on the generalization bounds of the form

$$|L(f) - \hat{L}(f)| \leq c_1 \mathbb{R}_N(\mathbb{F}) + c_2 \sqrt{\frac{\ln(\frac{1}{\delta})}{2N}}. \tag{8}$$

As the right hand side depends on the neural networks, which do not change in any substantial way, all that remains is understanding the multi-class loss.

To transform the outputs of the network into probabilities, the Softmax function is used

$$p_i = \frac{e^{-f_i(W;x)}}{\sum_{j=1}^{C} e^{-f_j(W;x)}}. \tag{9}$$

The cross-entropy loss is then defined simply as

$$\hat{L}(f) = -\sum_{n=1}^{N} \log \left(\frac{e^{-f_{y_n}(W;x)}}{\sum_{j=1}^{C} e^{-f_j(W;x)}} \right). \tag{10}$$

It's very easy to see that this reduces to the logistic loss in the binary case.

Classification now depends only on the margin $\eta_n = f_{y_n}(W;x) - \max_{j \neq y_n}\{f_j(W;x)\}$ – if $\eta_n > 0$ then the example is correctly classified. This means that, again, classification only cares about the sign of the margin and not the normalization of the neural network.

One final property of note is that for separable data, the loss monotonically decreases with increasing ρ. To see this, let us write $\alpha_{nj} = f_{y_n}(W;x) - f_j(W;x)$, which is a positive quantity in the separable case. Additionally define $g_n = -\log\left(\sum_{j \neq y_n} e^{-\alpha_{nj}}\right) = -\log\left(\sum_{j \neq y_n} e^{-\rho\tilde{\alpha}_{nj}}\right)$, which is clearly a monotonic function of ρ if all $\alpha_{nj} > 0$. We can now rewrite the Cross-entropy loss as

$$\hat{L}(f) = \sum_{n=1}^{N} \log(1 + e^{-g_n}), \tag{11}$$

which implies that we can drive the loss to 0 by increasing $\rho \to \infty$.

B Details of Figure 2 in the main text

Top The top left graph shows testing vs training cross-entropy loss for networks trained on the same data sets but with different initializations. The top right graph shows the testing vs training loss for the same networks, normalized by dividing each weight by the Frobenius norm of its layer. Notice that all points have zero classification error at training. The red point on the top right refers to a network trained on the same CIFAR data set but with randomized labels. It shows zero classification error at training and test error at chance level. The top line is a square-loss regression of slope 1 with positive intercept. The bottom line is the diagonal at which training and test loss are equal. The networks are 3-layer networks; the first layer is convolutional, 64 filters of size 5x5, stride 2, padding 2, no bias, ReLU activation; the second layer is also convolutional, 64 filters of size 5x5, stride 2, padding 2, no bias, ReLU activation; the third layer is fully connected, input size 64*8*8, output size 10, no bias, softmax activation. The training is on the CIFAR-10 dataset with 50k training examples, 10k testing examples. The network used for the point in red was trained similarly except the testing set and training set labels were randomized.

No data augmentation was performed, but data were normalized to mean (0.4914, 0.4822, 0.4465) and standard deviation (0.2023, 0.1994, 0.2010). Trained for 200 epochs with decreasing learning rate (0.01, 0.01, 0.001, 0.0001) until training error reached 0 percent. The weights were initialized

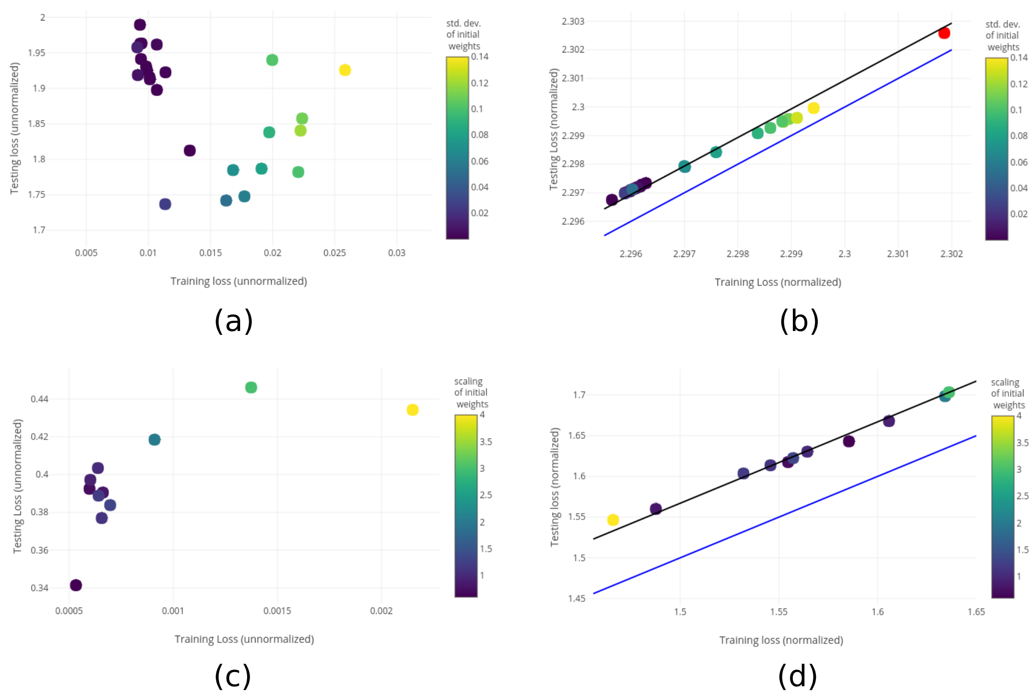


Figure 2 in the main text

from a normal distribution with mean 0 and standard deviation in [0.0001, 0.0002, 0.0003, 0.0004, 0.0005, 0.0006, 0.0007, 0.0008, 0.0009, 0.001, 0.0016, 0.0032, 0.0064, 0.0128, 0.0256, 0.0512, 0.06, 0.07, 0.08, 0.09, 0.1, 0.1024, 0.11, 0.12, 0.14]. For the red point (trained on random labels), the weights were initialized from a normal distribution with mean 0 and standard deviation 0.1.

Bottom The bottom graphs show similar experiments for ResNet-56 with 56 layers, demonstrating that our observations hold for state-of-the-art networks (testing errors ranging from 7% to 9%; on this 10-classes classification problem chance performance is 90% error). We again emphasize that the performance in classification of normalized and unnormalized networks is exactly the same. The normalization in the case of ResNet was performed by using the norm of the output of each network on one example from CIFAR-10, because we found it computationally difficult to directly evaluate the effective norm of the residual layers.

The networks were trained for 200 epochs with learning rate = 0.01 for the first 100 epochs and learning rate = 0.001 for the second 100 epochs. SGD was used with batch size of 128 and shuffled training data with random crop and horizontal flip data augmentation.

C Selecting minimizers with same training performance but different test performance

First we start with a common observation: even when two networks have the same architecture, same optimization meta parameters and same training loss, they usually have different test performances (i.e. error and loss), because the stochastic nature of the minimization process leads to convergence to different minima among the many existing in the loss landscape Poggio & Liao (2017); Poggio et al. (2017; 2018).

With standard settings the differences are usually small (though significant, as shown later). We use therefore two approaches to magnify the effect:

- Initialize networks with different levels of "random pretraining": the network is pretrained for a specified number of epochs on "corrupted" training data — the labels of a portion of the examples are swapped with each other in a random fashion.

- Initialize the weights of the networks with different standard deviations of a diagonal Gaussian distribution. As it turns out, different standard deviations yield different test performance.

Similar techniques have been used previously Neyshabur et al. (2017); Liang et al. (2017). We show the results of "random pretraining" with networks on CIFAR-10 (Figure 5) and CIFAR-100 (Figure 7) and initialization with different standard deviations on CIFAR-10 (Figure 6) and CIFAR-100 (Figure 8).

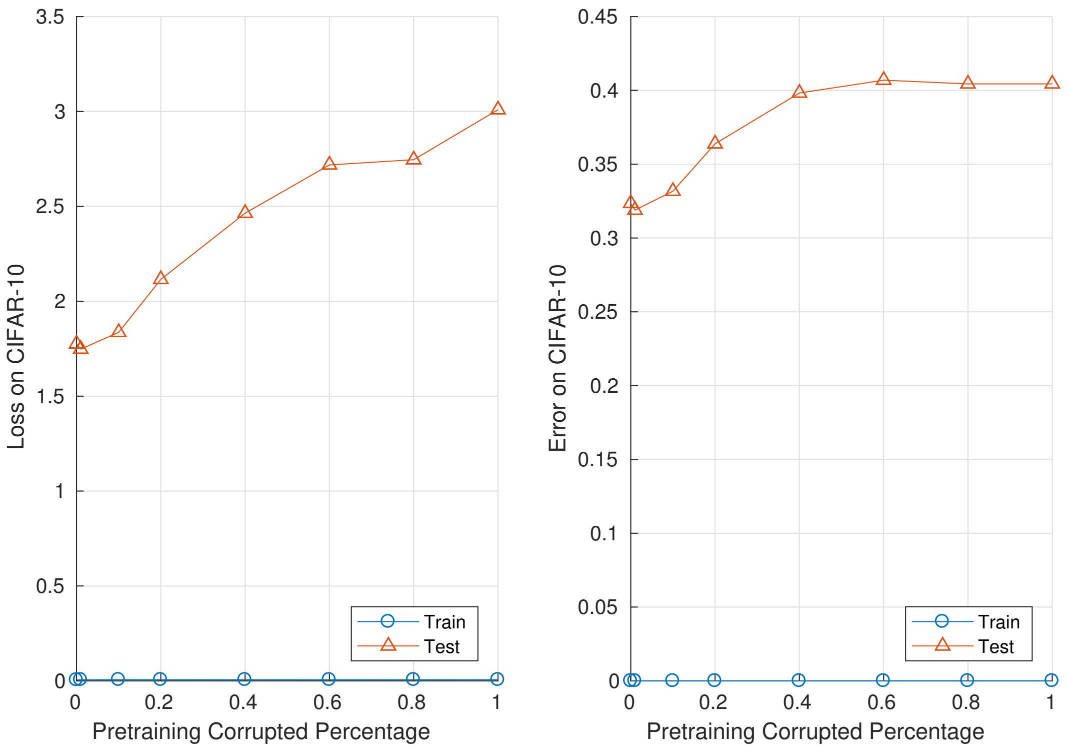
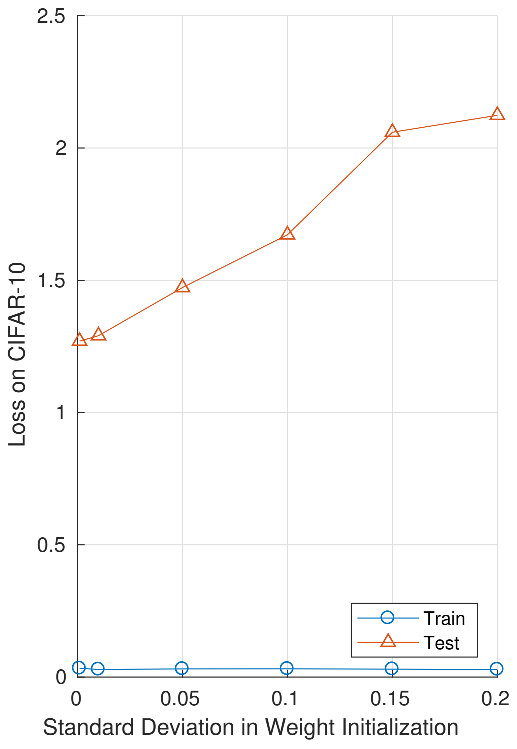


Figure 5: *Random Pretraining vs Generalization Performance on CIFAR-10: a 5-layer ConvNet (described in section L.2) is pretrained on training data with partially "corrupted" labels for 30 epochs. It is then trained on normal data for 80 epochs. Among the network snapshots saved from all the epochs we pick a network that is closest to an arbitrarily (but low enough) chosen reference training loss (0.006 here). The number on the x axis indicates the percentage of labels that are swapped randomly. Loss is cross-entropy loss; error is classification error. As pretraining data gets increasingly "corrupted", the generalization performance of the resultant model becomes increasingly worse, even though they have similar training losses and the same zero classification error in training. Batch normalization (BN) is used. After training, the means and standard deviations of BN are "absorbed" into the network's weights and biases. No data augmentation is performed.*



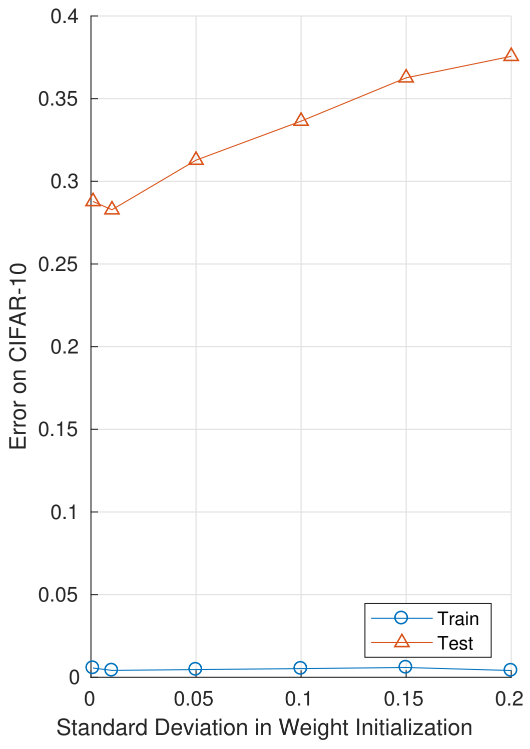


Figure 6: *Standard Deviation in weight initialization vs generalization performance on CIFAR-10: the network is initialized with weights of different standard deviations. The other settings are the same as in Figure 5. As the norm of the initial weights becomes larger, the generalization performance of the resulting model is worse, even though all models have the same classification error in training.*

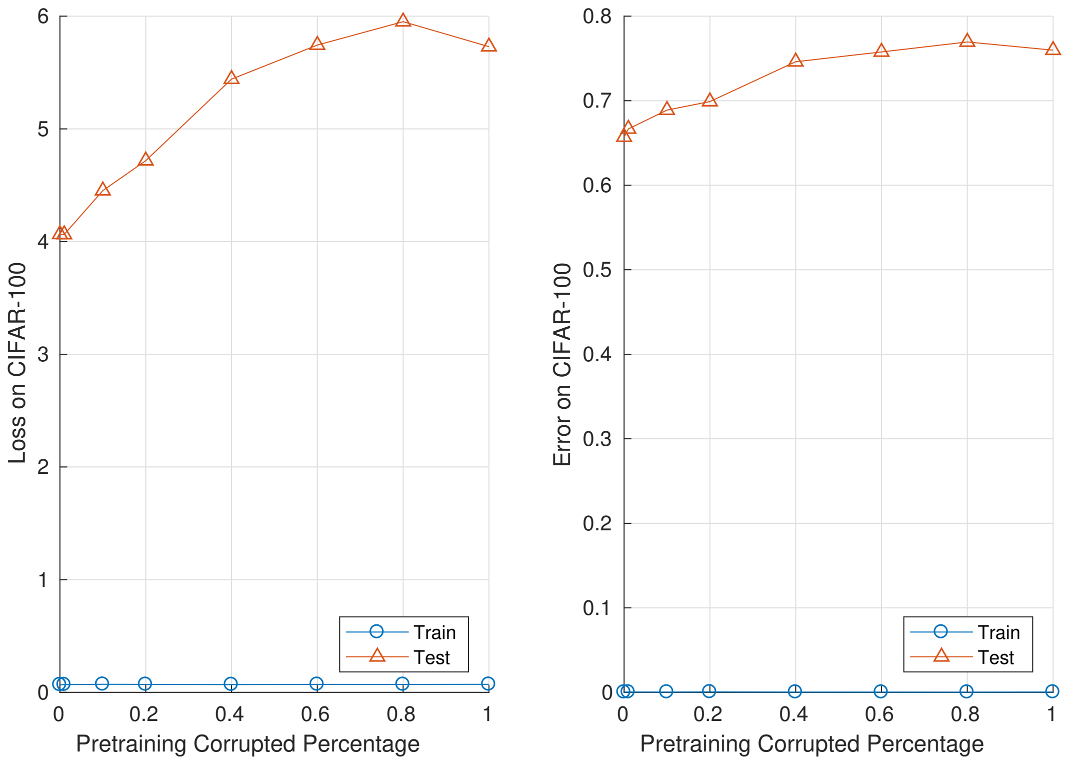


Figure 7: *Same as Figure 5, but on CIFAR-100.*

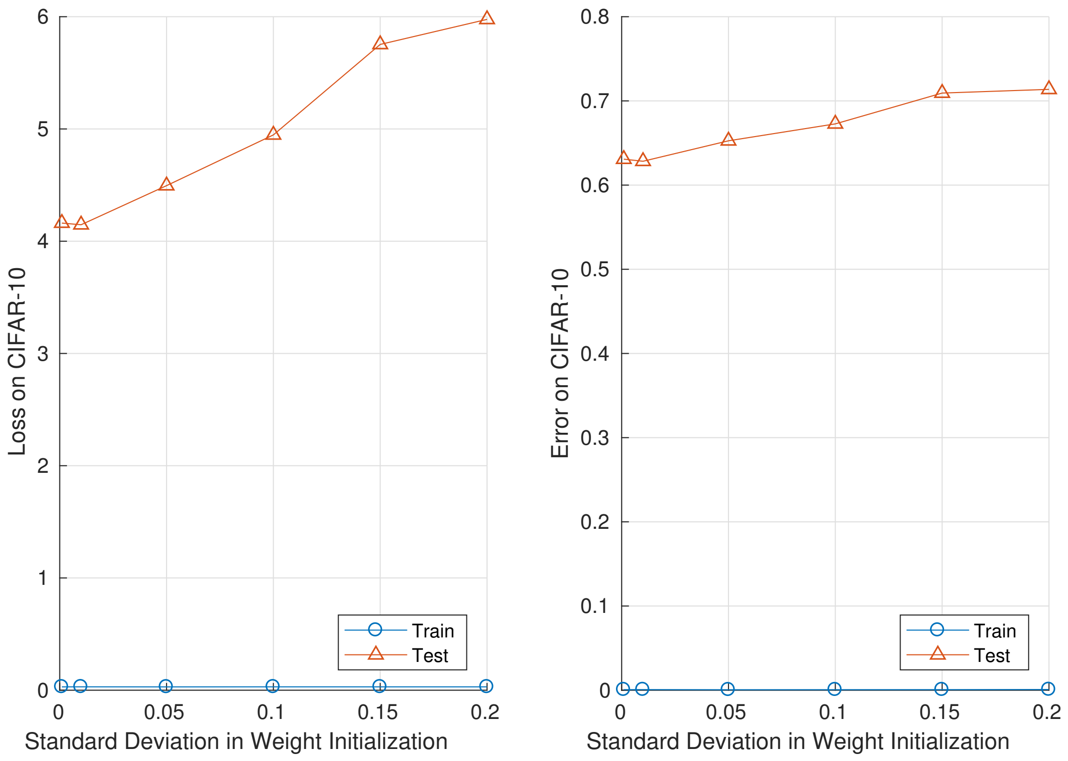


Figure 8: *Same as Figure 6, but on CIFAR-100.*

D Linear relations between training and test loss of normalized networks

Our observations are that the linear relationship between train loss and test loss for normalized networks hold in a robust way under several conditions:

- *Independence from Initialization:* The linear relationship is independent of whether the initialization is via pretraining on randomly labeled natural images or whether it is via larger initialization, as shown by Figures 11 and 12.

- *Independence from Network Architecture:* The linear relationship of the test loss and train loss does not depend on the network architectures we tried. Figure 10, shows the linear relationship for a 3 layer network without batch normalization while Figures 11, 12 show the linear relationship for a 5 layer network with batch normalization on CIFAR10. Additional evidence can be found in Figures 9. Figures in the main text show the same linear relationship for ResNet architectures.

- *Independence from Data Set:* Figures 10, 11, 12, 9 show the linear relationship on CIFAR10 while Figures 17 and 18 show the linear relationship on CIFAR100.

- *Norm independence:* Figures 9 show that the choice of L_p norm used for normalization does not matter – as expected, see main text. The constants underlying the equivalence of different norms depend on the dimensionality of the vector spaces.

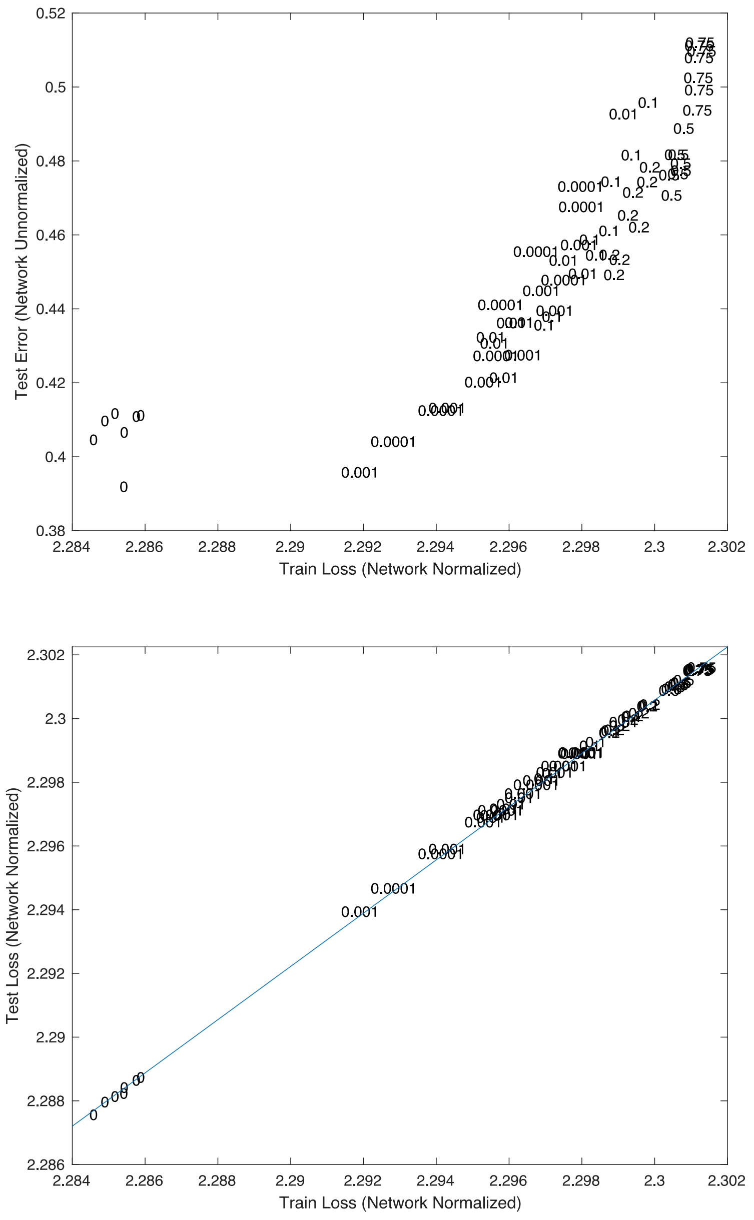


Figure 9: *Test loss/error vs training loss with all networks normalized layerwise by the L1 norm (divided by* 100 *to avoid numerical issues because the L1 norms are here very large). The model was a 3 layer neural network described in section L.1.1 and was trained with 50K examples on CIFAR10. The networks were normalized after training each up to epoch* 300 *and thus different points on the plot correspond to different training losses. Figure 10 shows that the normalized network does not depend on the value of the training loss before normalization. The slope and intercept of the line of best fit are* 0.8358 *and* 0.3783 *respectively. The ordinary and adjusted* R^2 *values are both* 0.9998 *while the root mean square (RMSE) was* 5.6567×10^{-5}*. The numbers in the figure indicate the amount of corruption of random labels used during the pretraining.*

- *Normalization is independent of training loss:* Figure 10 shows that networks with different cross-entropy training losses (which are sufficiently small to guarantee zero classification error), once normalized, show the same linear relationship between train loss and test loss.

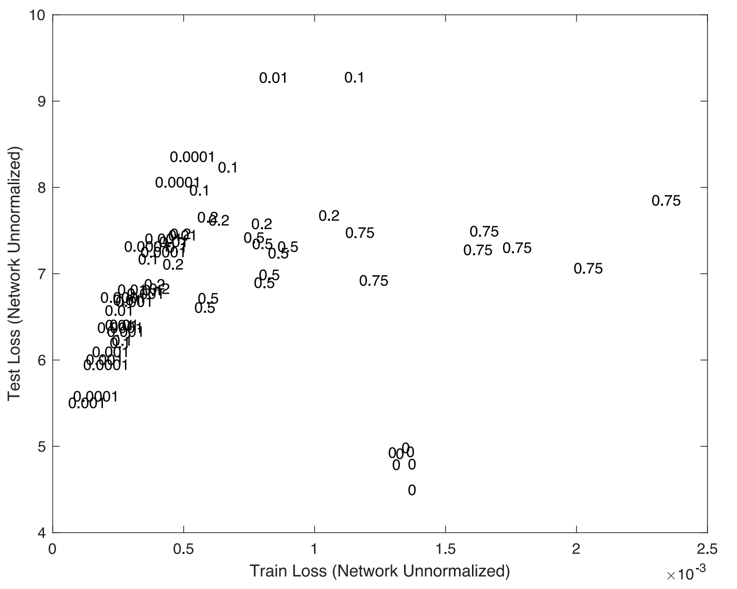
E Higher Capacity leads to higher Test Error

One way to formalize the upper bound on classification error by the logistic loss is to consider the excess classification risk $R(f) - R^*$, where $R(f)$ is the classification error associated with f and R^* is the Bayes error Bartlett et al. (2003). Let us call as before $L(f)$ the expected cross-entropy loss and L^* the optimal expected cross entropy loss. Then the following bound holds for binary classification in terms of the so-called ψ-transform of the logistic loss ℓ:

$$R(f) - R^* \leq \psi^{-1}(L(f) - L^*) \tag{12}$$

where the ψ function for the logistic is similar to the ψ for the exponential loss which is $\psi(x) = 1 - \sqrt{1 - x^2}$. The key point here is that ψ^{-1} is *monotonic*: minimizing the logistic or cross-entropy loss implies minimizing the classification error.

Our arguments imply that among two unnormalized minimizers of the exponential loss that achieve the same given small empirical loss $\hat{L} = \epsilon$, the minimizer with higher product of the norms $\rho_1, \cdots, \rho_K$ has the higher capacity and thus the highest expected loss L. Experiments support this claim, see Figure 13. Notice the linear relationship of test loss with increasing capacity on the top right panels of Figure 11, 12, 17, 18.



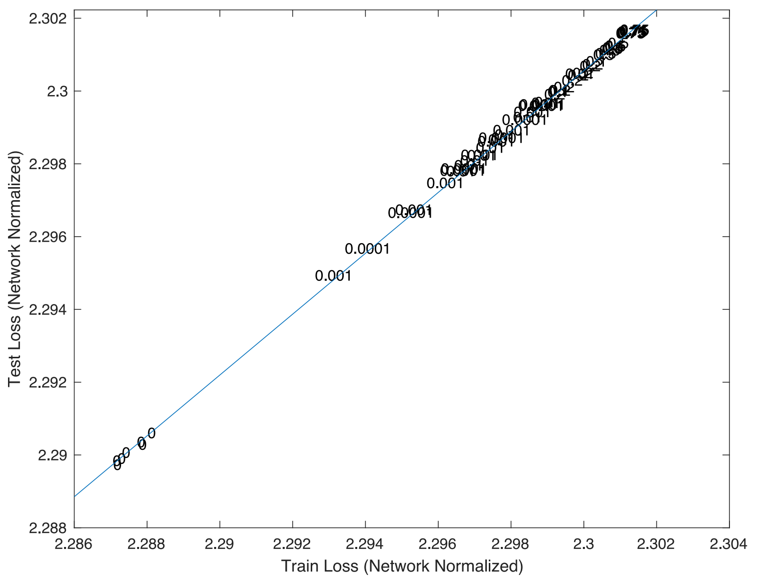
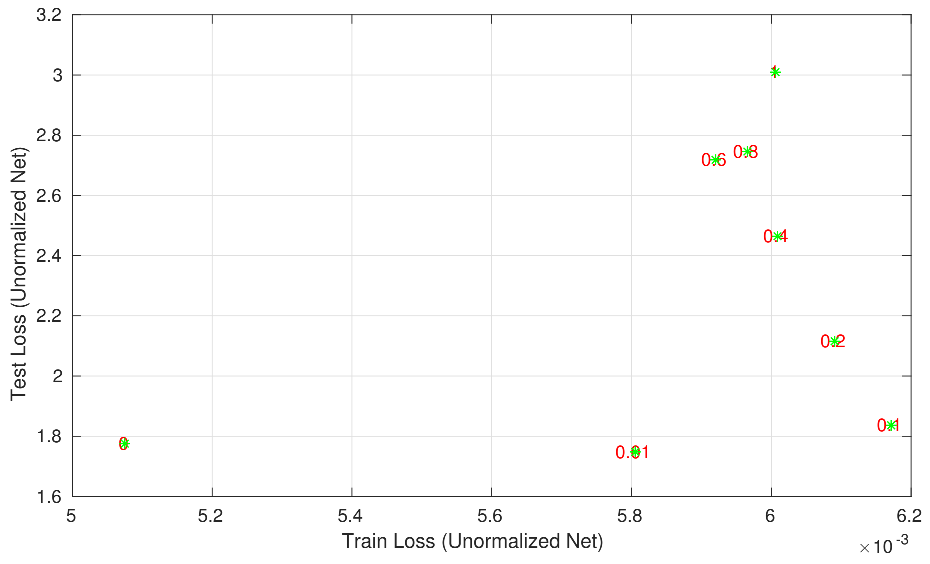


Figure 10: *Left: test loss vs training loss with all networks normalized layerwise by the Frobenius norm. Right: test loss vs training loss with all unnormalized networks. The model was a 3 layer neural network described in section L.1.1 and was trained with 50K examples on CIFAR10. In this experiments the networks converged (and had zero train error) but not to the same loss. All networks were trained for* 300 *epochs. The losses range approximately from* 1.5×10^{-4} *to* 2.5×10^{-3}. *The numbers in the figure indicate the amount of corruption of random labels used during pretraining. The slope and intercept of the line of best fit are* 0.836 *and* 0.377 *respectively. The ordinary and adjusted* R^2 *values are both* 0.9998 *while the root mean square (RMSE) is* 4.7651×10^{-5}.



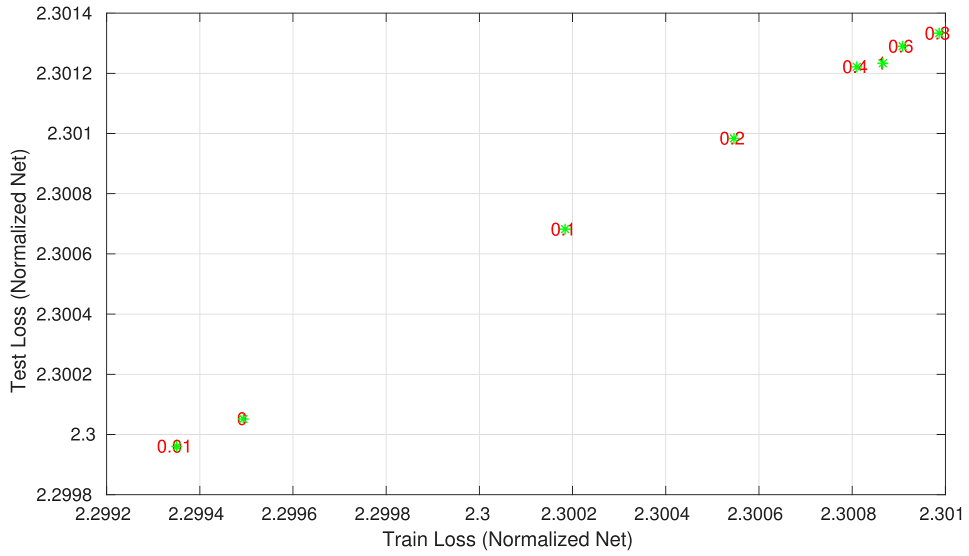
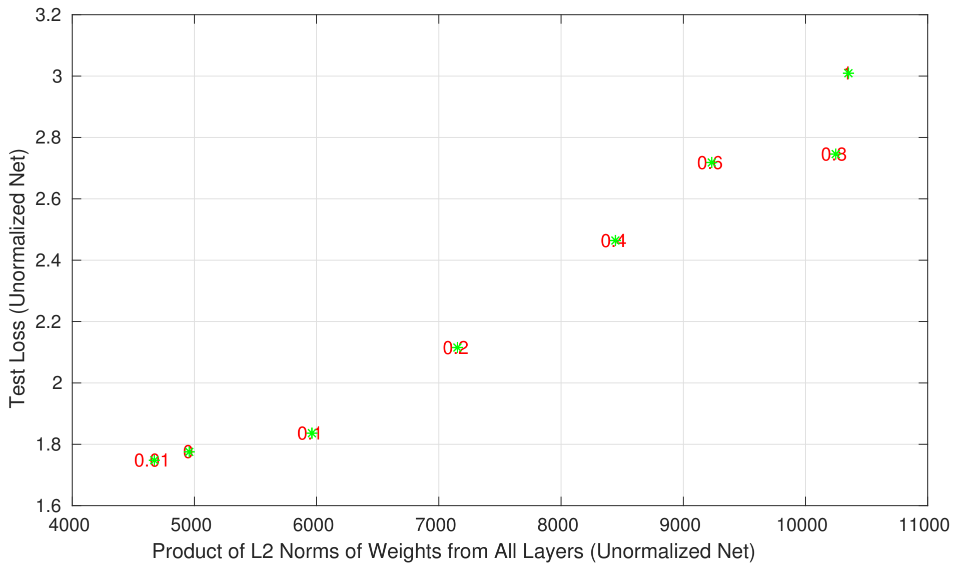


Figure 11: *(Part 1) Random pretraining experiment with batch normalization on CIFAR-10 using a 5 layer neural network as described in section L.2. The red numbers in the figures indicate the percentages of "corrupted labels" used in pretraining. The green stars in the figures indicate the precise locations of the points.* ***Top:*** *Training and test losses of unnormalized networks: there is no apparent relationship.* ***Bottom:*** *under layerwise normalization of the weights (using the Frobenius norm), there is a surprisingly good linear relationship between training and testing losses, implying tight generalization. See next page for the product of L2 norms and classification errors.*



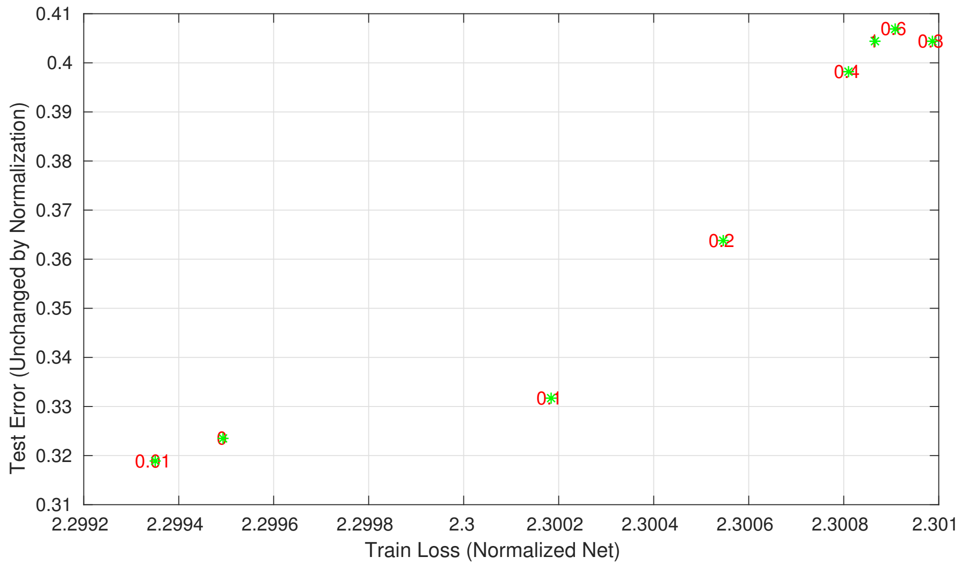
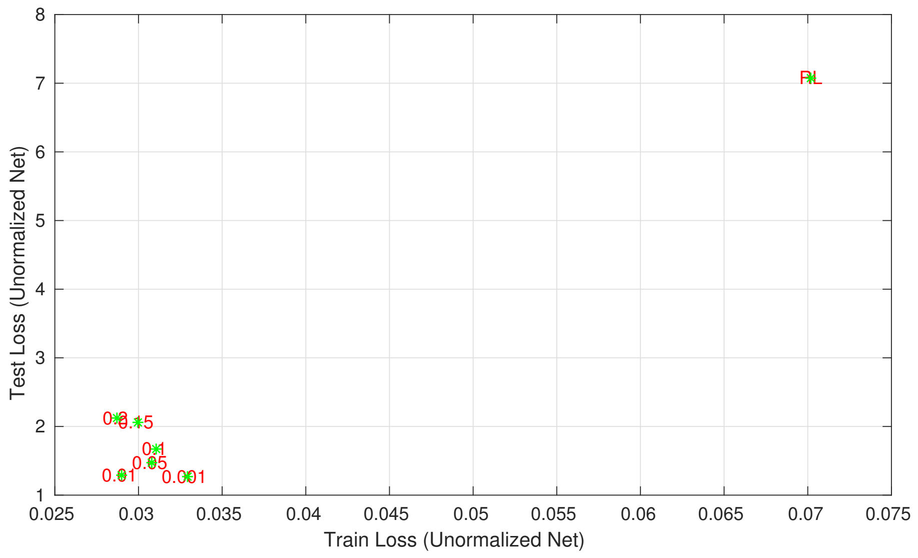


Figure 11: *(Part 2)* ***Top*** *the product of L2 norms from all layers of the network. We observe a positive correlation between the norm of the weights and the testing loss.* ***Bottom:*** *under layerwise normalization of the weights (using the Frobenius norm), the classification error does not change.*



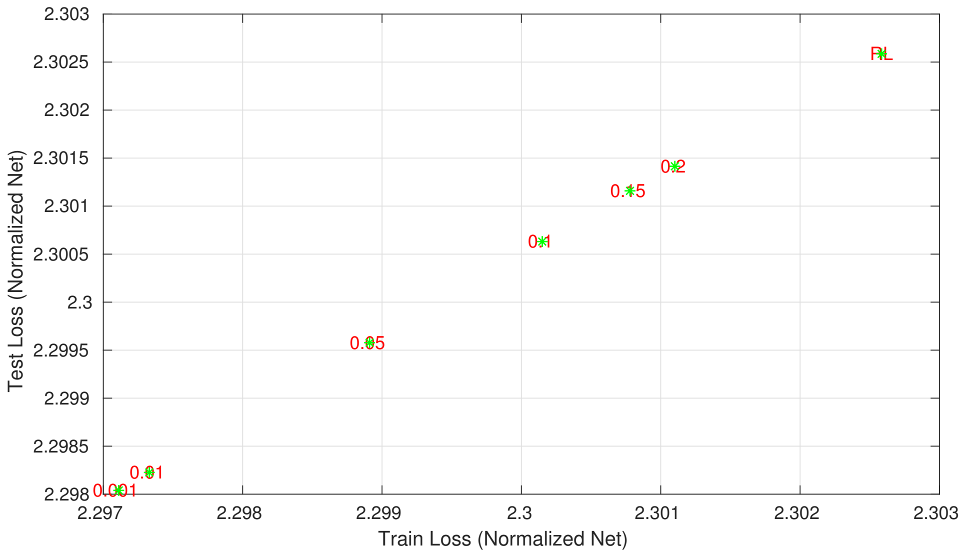
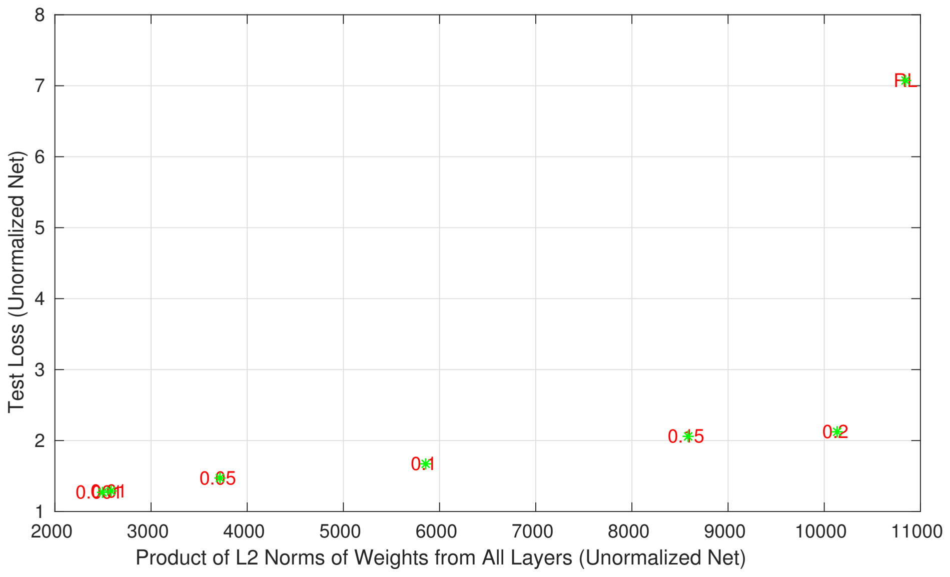


Figure 12: *(Part 1) Same as Figure 11 but using different standard deviations for initialization of the weights instead of "random pretraining". The red numbers in the figures indicate the standard deviations used in initializing weights. The "RL" point (initialized with standard deviation 0.05) refers to training and testing on completely random labels.*



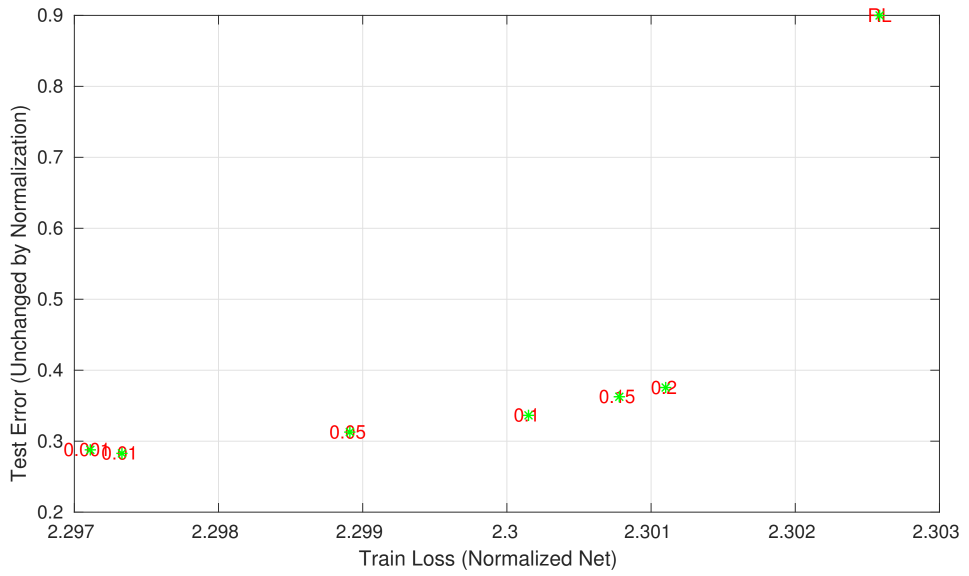


Figure 12: *(Part 2) Same as Figure 11 but using different standard deviations for initialization of the weights instead of "random pretraining". The red numbers in the figures indicate the standard deviations used in initializing weights. The "RL" point (initialized with standard deviation 0.05) refers to training and testing on completely random labels.*

Figure 13 shows that when the capacity of a network (as measured by the product norm of the layers) increases, so does the test error.

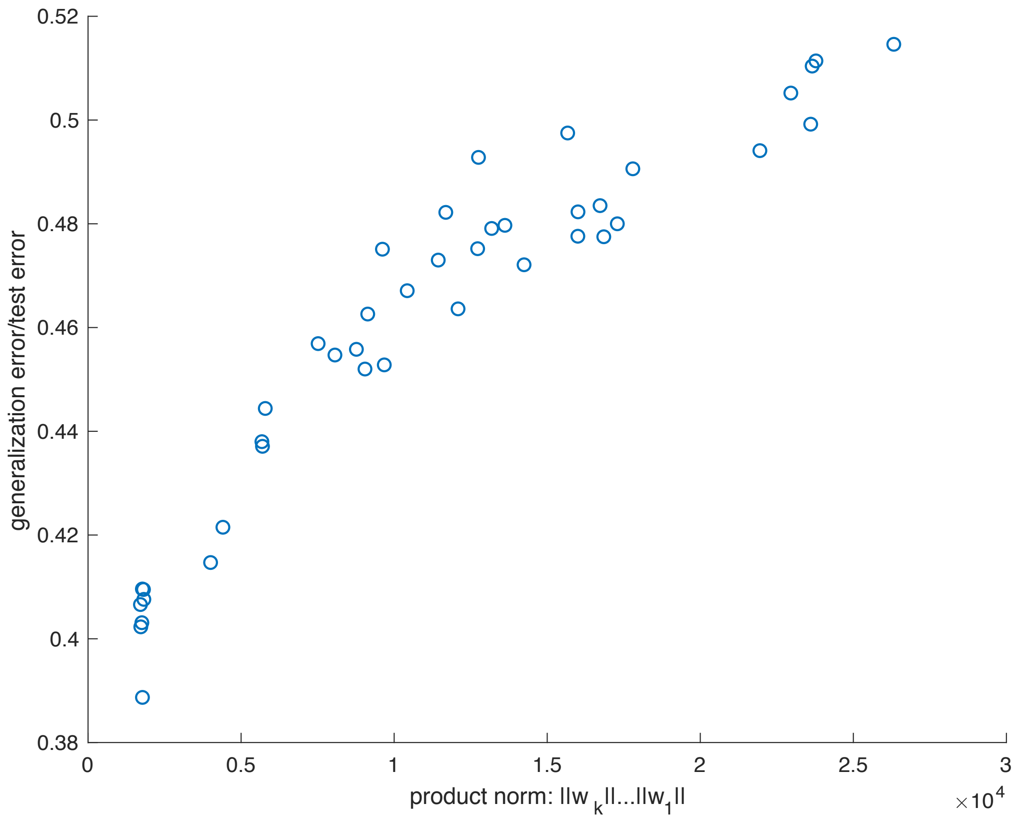


Figure 13: *Plot of test error vs the product of the Frobenius norms of the layers* $\|W\|_{product} = \prod_{l=1}^{L} \|W_l\|$. *The model was a 3 layer neural network described in section L.1.1 and trained with 50K examples on CIFAR10. The models were obtained by pretraining on random labels and then fine tuning on natural labels. SGD without batch normalization was run on all networks in this plot until each reached approximately* 0.0044 ± 0.0001 *cross-entropy loss on the training data. Similar results can be found in Neyshabur et al. (2017); Liang et al. (2017) .*

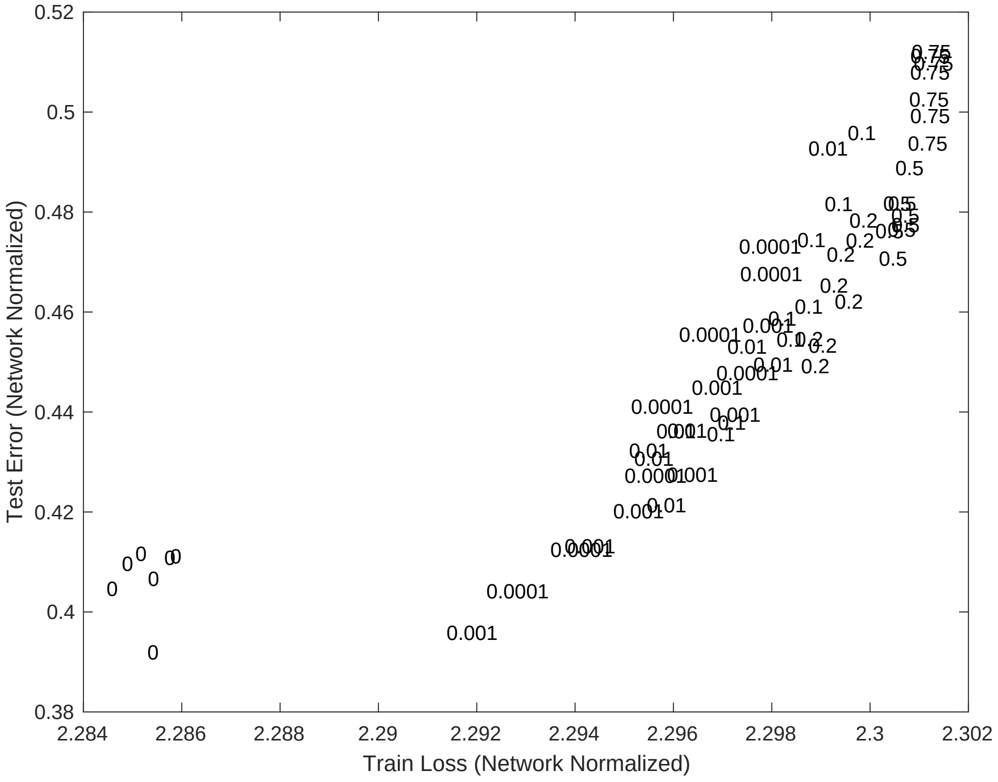
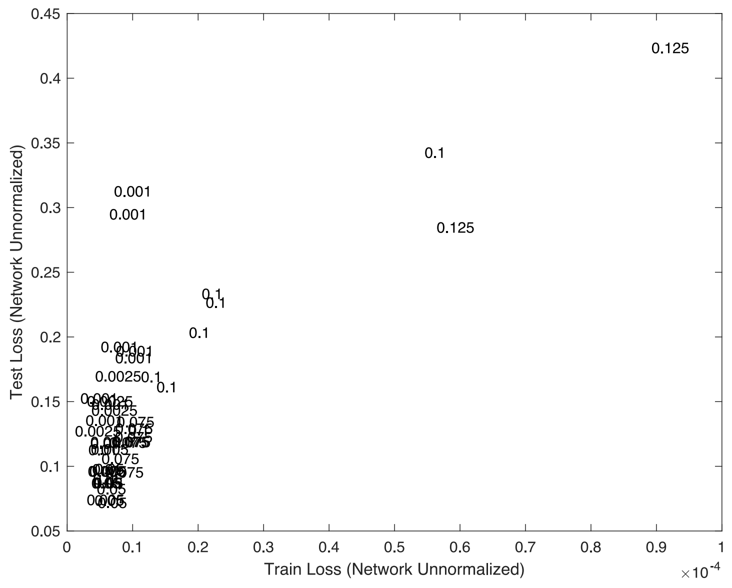


Figure 14: *Test classification error vs training cross-entropy loss with all networks normalized layerwise. The model was a 3 layer neural network and was trained with 50K examples on CIFAR10. The empirical dependence of classification error on normalized cross-entropy loss shown here is different from the predicted Bartlett et al. (2003) upper bound provided by* ψ^{-1} *(pers. comm. Y. Yao).*

F Results on MNIST

This section shows figures replicating the main results on the MNIST data set. Figures 15 and 16 show that the linear relationship holds after normalization on the MNIST data set. Figure 16 shows the linear relationship holds after adding the point trained only on random labels.



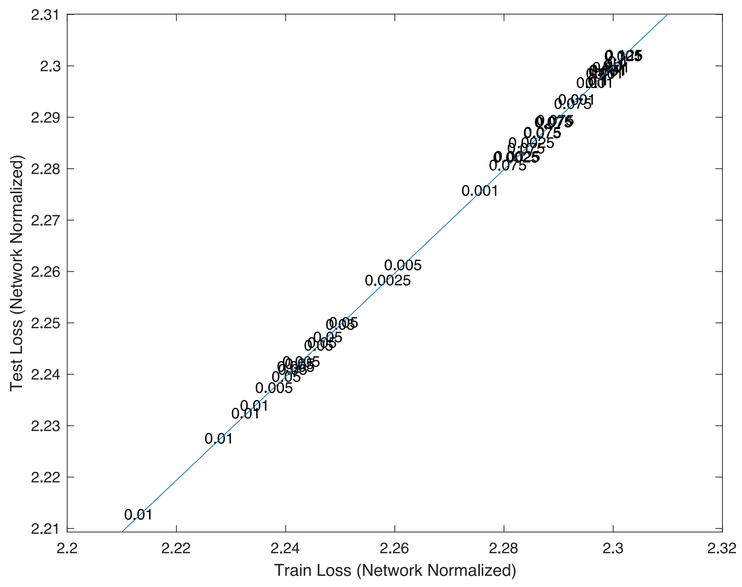
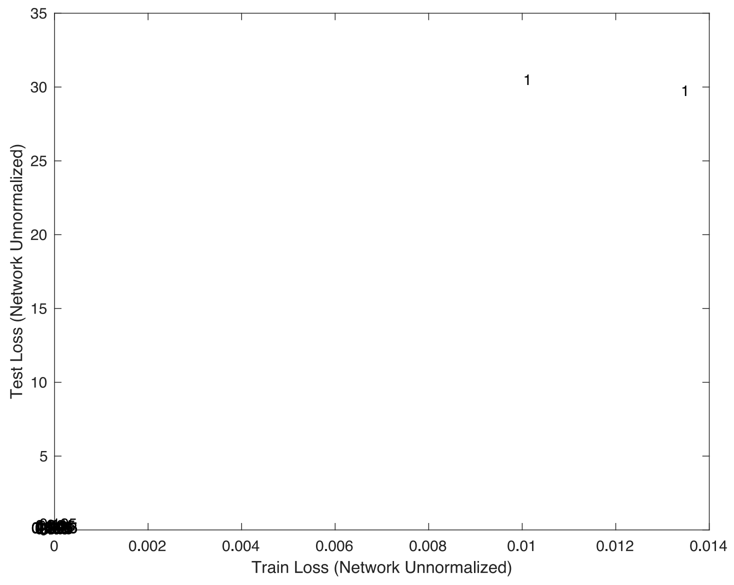


Figure 15: *The figure shows the cross-entropy loss on the test set vs the training loss for networks normalized layerwise in terms of the Frobenius norm. The model was a 3 layer neural network described in section L.1.2 and was trained with 50K examples on MNIST. All networks were trained for* 800 *epochs. In this experiment the networks converged (and had zero train error) but not to the same loss. The slope and intercept of the line of best fit are* 1.0075 *and* -0.0174 *respectively. The ordinary and adjusted* R^2 *values are both* 1.0000 *while the root mean square (RMSE) was* 9.1093×10^{-4}. *The makers indicate the size of the standard deviation of the normal used for initialization.*



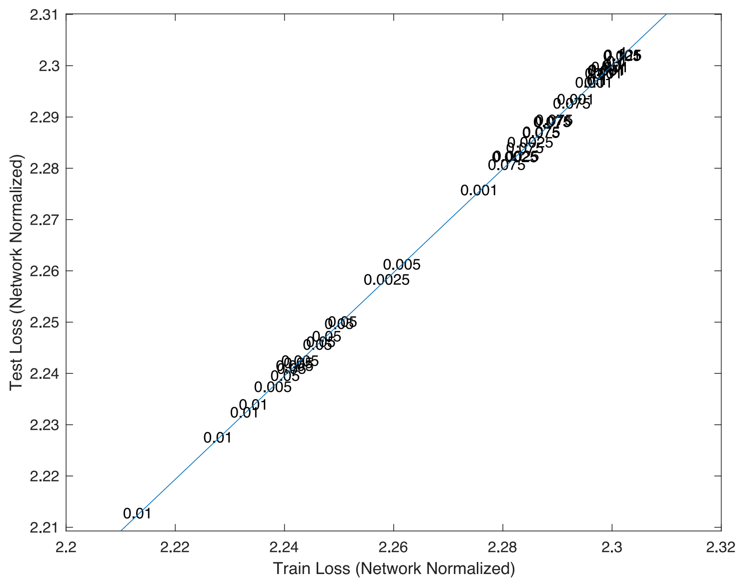
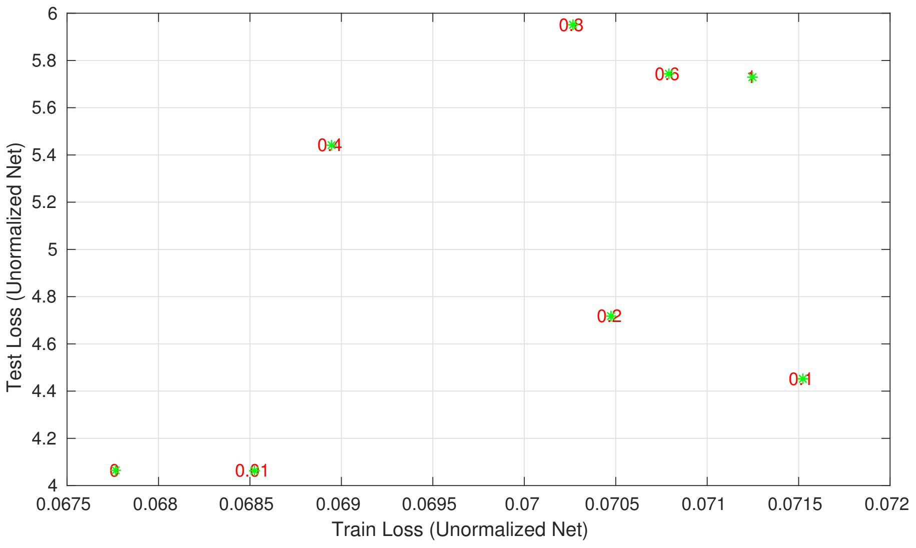


Figure 16: *The figure shows the cross-entropy loss on the test set vs the training loss for networks normalized layerwise in terms of the Frobenius norm. The model was a 3 layer neural network described in section L.1.1 and was trained with 50K examples on CIFAR10. All networks were trained for* 800 *epochs. In this experiment the networks converged (and had zero train error) but not to the same loss. The slope and intercept of the line of best fit are* 1.0083 *and* -0.0191 *respectively. The ordinary and adjusted* R^2 *values are both* 1.0000 *while the root mean square (RMSE) was* 9.1093×10^{-5}. *The points labeled* 1 *were trained on random labels; the training loss was estimated on the same randomly labeled data set. The points marked with values less than* 1 *were only trained on natural labels and those makers indicate the size of the standard deviation of the normal used for initialization.*

G Results on CIFAR-100

This section is about replicating the main results on CIFAR-100. Figure 7 shows how different test performance can be obtained with pretraining on random labels while Figure 8 shows that different increasing initializations are also effective.

Figures 17 and 18 show that the linear relationship holds after normalization, regardless of whether the training was done with pretraining on random labels or with large initialization.



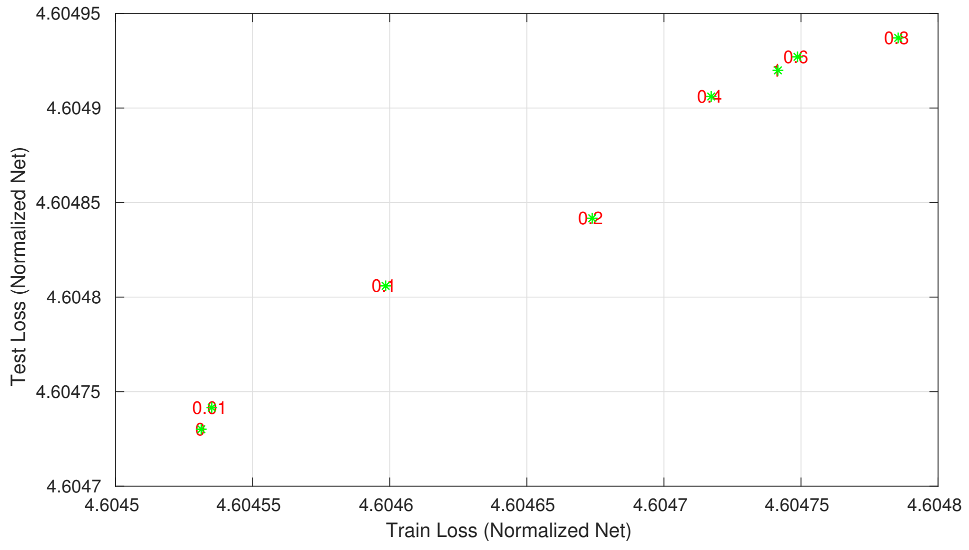
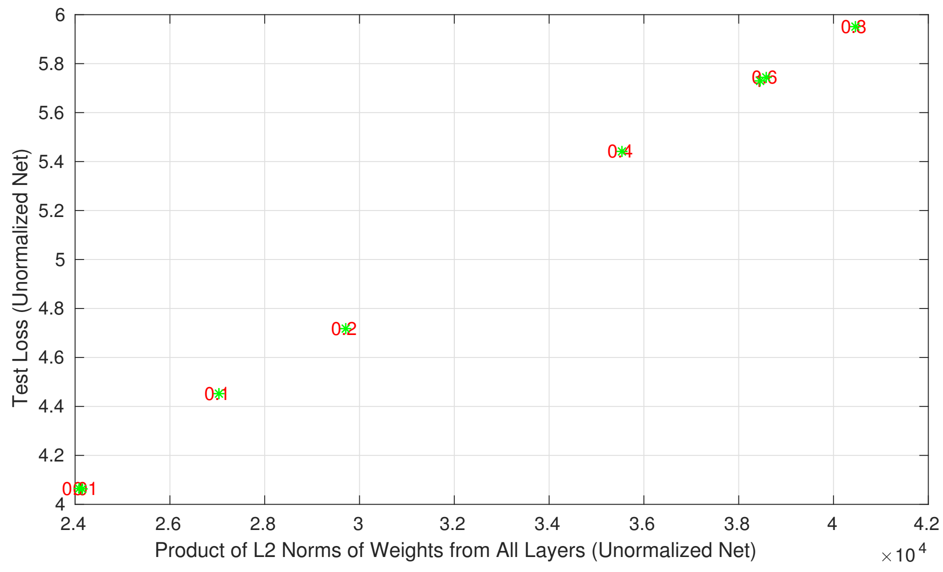


Figure 17: *(Part 1) Same as Figure 11 but on CIFAR-100.*



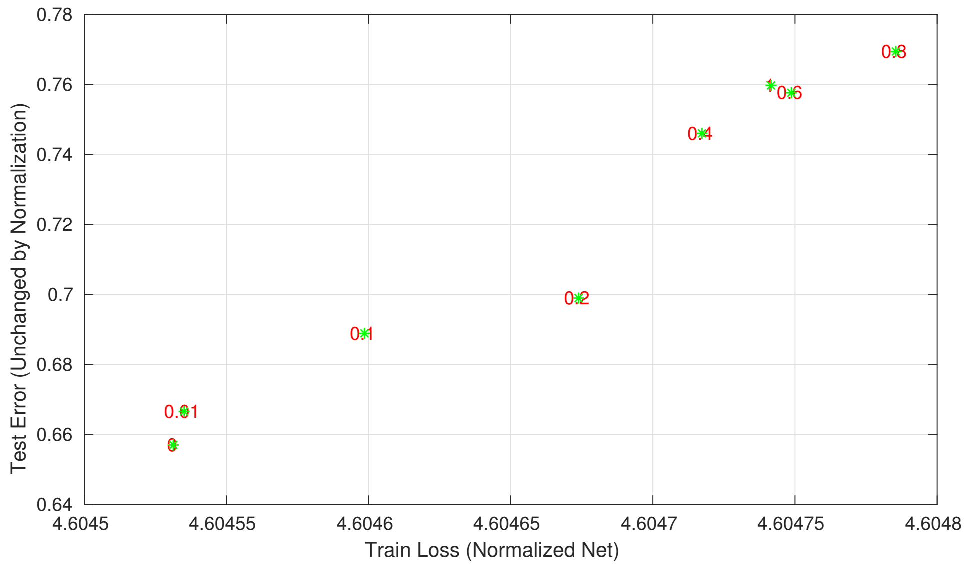
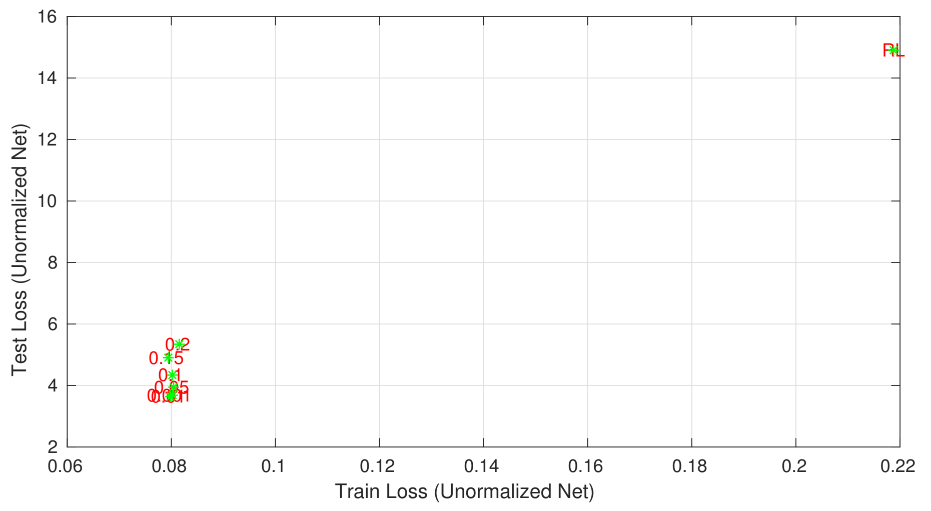


Figure 17: *(Part 2) Same as Figure 11 but on CIFAR-100*



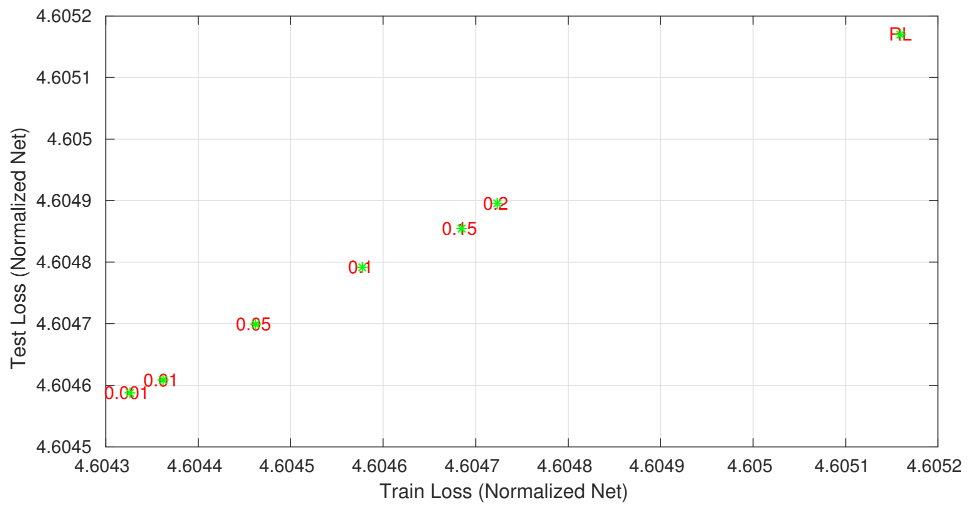
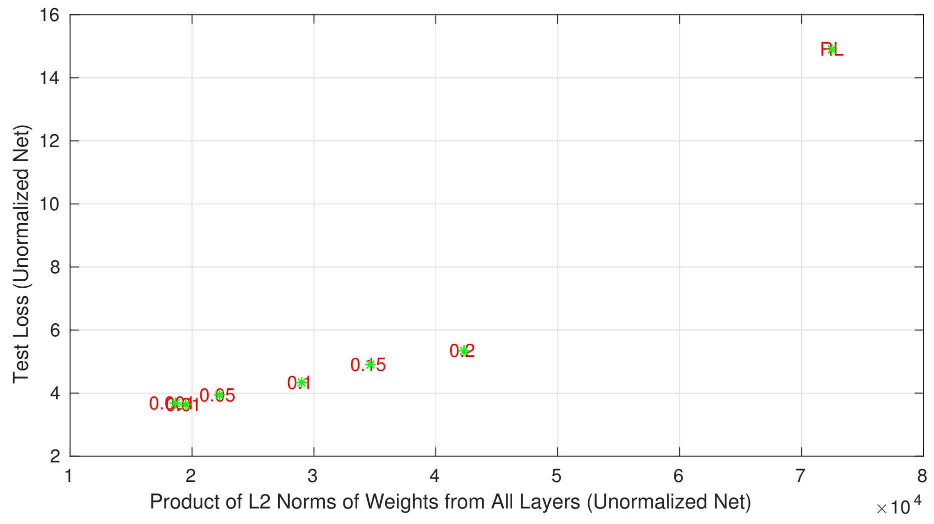


Figure 18: *(Part 1) Same as Figure 12 but on CIFAR-100.*



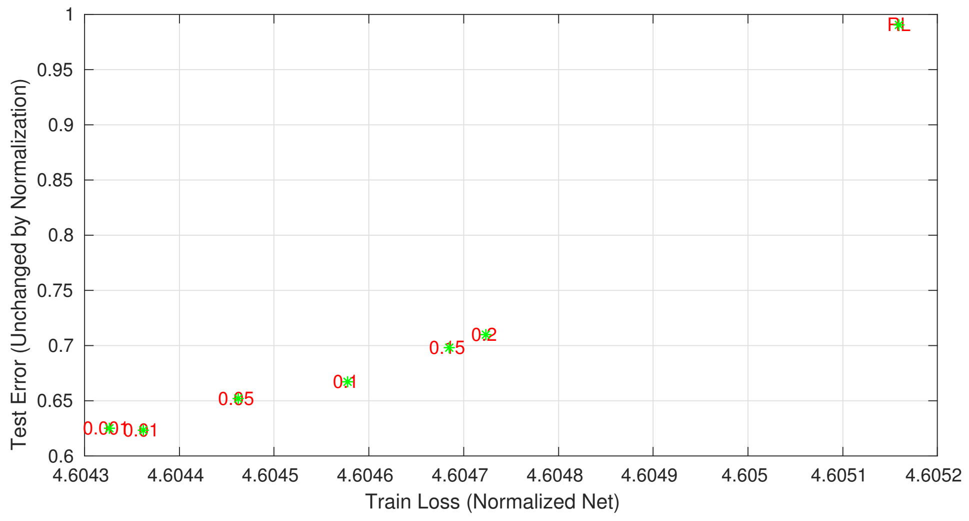


Figure 18: *(Part 2) Same as Figure 12 but on CIFAR-100.*

H Results with ResNet architectures

This section is about replicating the main results with ResNets (see main text figures). To avoid using the product of layerwise norms – whose form is not obvious for ResNets – we normalized the different architectures using their output on the same single image, exploting the fact that all the networks only differ in their initialization (for this reason we did not plot the RL point because this shortcut to normalization does not apply in the case of a different training set with random labels).

I Generalization with $N \to \infty$

While in this article we use the term generalization when the offset in the difference between training and test losses is small, the technical definition of "generalization" just requires that the offset decreases with increasing N. This means that for all $f \in \mathbb{F}$, $\lim_{N\to\infty} |L(\tilde{f}) - \hat{L}(\tilde{f})| \to 0$, since in general we expect the Rademacher complexity to decrease as N increases. As Figure 19 shows, $|L(\tilde{f}) - \hat{L}(\tilde{f})|$ does in fact decrease with increasing N (notice that $L(\tilde{f}) \approx \hat{L}(\tilde{f})$ around $N = 50000$ for normalization to the unit L_2 ball of our network). These arguments do not depend on the specific form of the Rademacher complexity. For deep networks several measures of complexity have been proposed – mostly involving products of layer-wise normsAnthony & Bartlett (2002); Neyshabur et al. (2017); Bartlett et al. (2017); Liang et al. (2017).

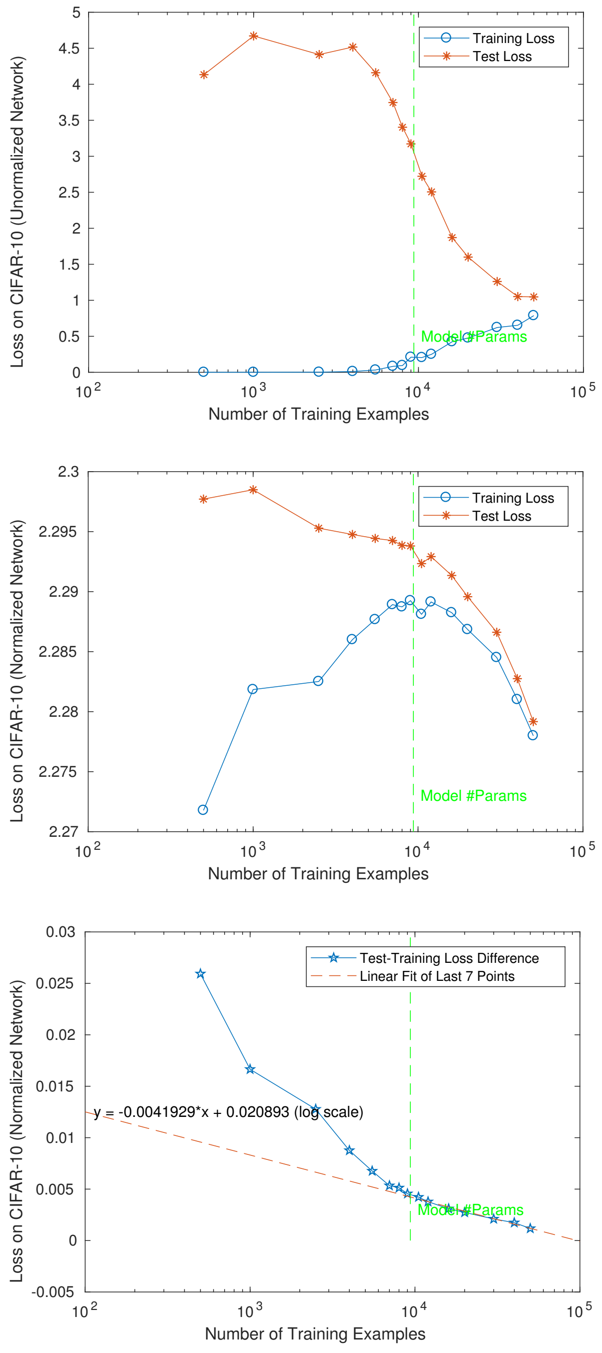


Figure 19: *(Part 1)* ***Top:*** *Unnormalized cross-entropy loss in CIFAR10 for normally labeled data.* ***Middle:*** *Cross-entropy loss for the normalized network for normally labeled data.* ***Bottom:*** *Generalization cross-entropy loss (difference between training and testing loss) for the normalized network for normally as a function of the number of data N. The generalization loss converges to zero as a function of N but very slowly.*

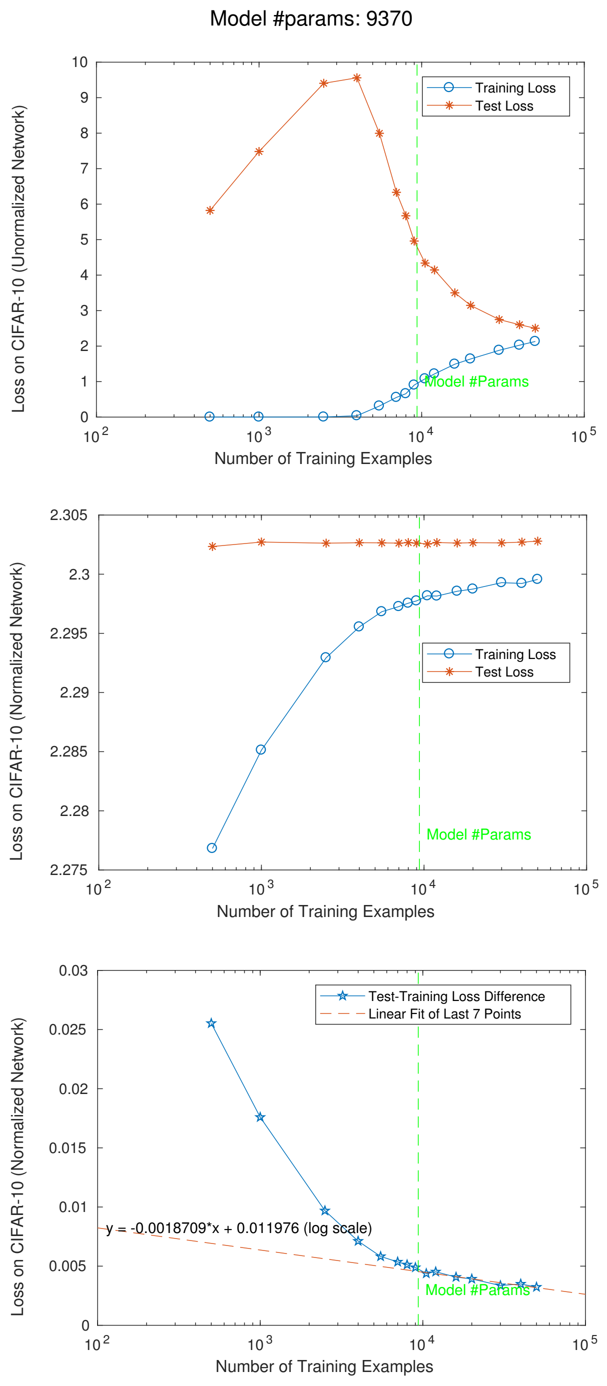


Figure 19: *(Part 2)* ***Top:*** *Unnormalized cross-entropy loss in CIFAR10 for randomly labeled data.* ***Middle:*** *Cross-entropy loss for the normalized network for randomly labeled data.* ***Bottom:*** *Generalization cross-entropy loss (difference between training and testing loss) for the normalized network for randomly labeled data as a function of the number of data N. The generalization loss converges to zero as a function of N but very slowly.*

J Margin bounds

A typical margin bound for classification Bartlett & Shawe-Taylor (1998) is

$$|L_{binary}(f) - L_{surr}(f)| \leq b_1 \frac{\mathbb{R}_N(\mathbb{F})}{\eta} + b_2 \sqrt{\frac{\ln(\frac{1}{\delta})}{2N}} \tag{13}$$

where η is the margin, $L_{binary}(f)$ is the expected classification error, $L_{surr}(f)$ is the empirical loss of a surrogate loss such as the logistic or the exponential. For a point x, the margin is $\eta \sim y\rho\tilde{f}(x)$. Since $\mathbb{R}_N(\mathbb{F}) \sim \rho$, the margin bound says that the classification error is bounded by $\frac{1}{\eta}$ that is by the value of $\tilde{f}$ on the "support vectors" – that is on the x_i, y_i s.t $\arg\min_n y_n \tilde{f}(x_n)$.

We have looked at data showing the test classification error versus the inverse of the margin. The data are consistent with the margin bound in the sense that the error increases with the inverse of the margin and can be bounded by a straight line with appropriate slope and intercept. Since the bound does not directly provide slope and intercept, the data do not represent a very convincing evidence.

K Numerical values of normalized loss

Why are all the cross-entropy loss values close to chance (e.g. $\ln 10 \approx 2.3$ for a 10 class data set) in the plots for convnets – bit not for ResNets – showing the linear relationship? This is because most of the (correct) outputs of the normalized neural networks are close to zero as shown by Figure 20. We would expect the norm of the network to be appromimately bounded by $|\tilde{f}(\tilde{W};x)| \lessapprox |\tilde{W}||x| = |x|$; the data x is usually pre-processed to have mean 0 and a standard deviation of 1. In fact, for the MNIST experiments, the average value $f(x)$ of the most likely class according to the normalized neural network is 0.026683 with a standard deviation 0.007144. This means that significant differences, directly reflecting the predicted class of each point, are between 0.019539 and 0.033827. This in turn implies that the exponentials in the cross-entropy loss are all very close to 1. Before normalization (which of course does not affect the classification performance), the average value $f(x)$ of the most likely class was 60.564373 with standard deviation 16.214078.

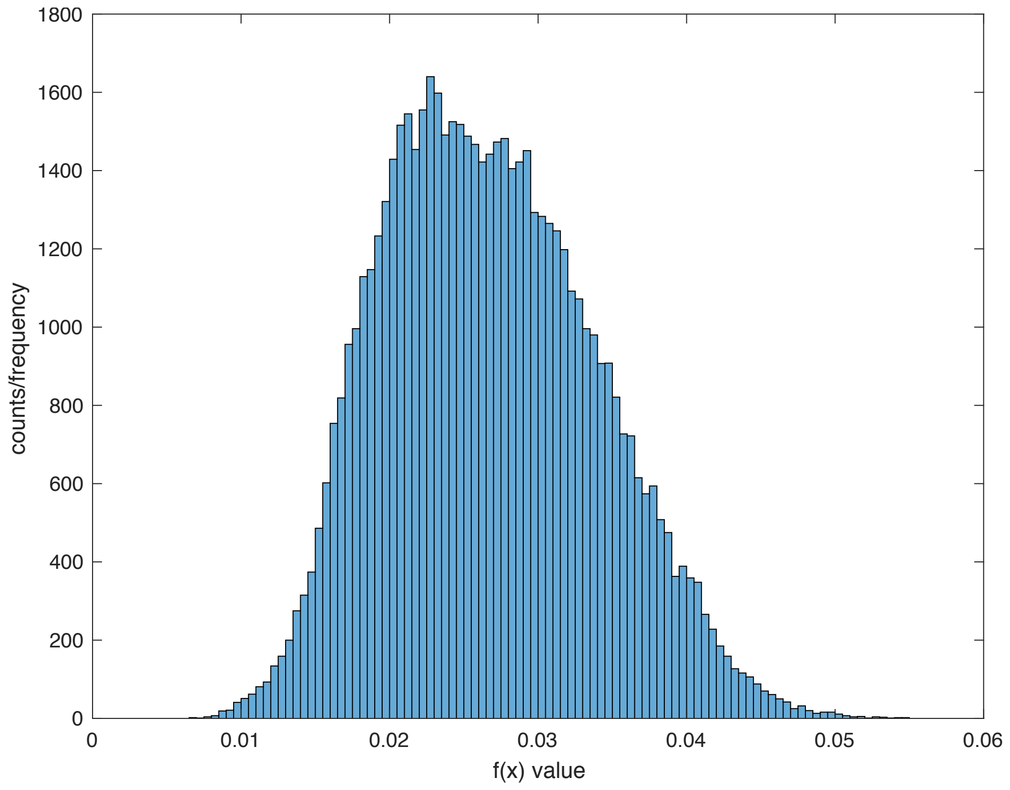


Figure 20: *Histogram of the values of $\tilde{f}(x)$ for the most likely class of the layerwise normalized neural network over the 50K images of the MNIST training set. The average value $\tilde{f}(x)$ of the most likely class according to the normalized neural network is* 0.026683 *with standard deviation* 0.007144*.*

L Deep Neural Network Architecture

L.1 Three layer network

L.1.1 Network with 24 filters

The model is a 3-layer convolutional ReLU network with the first two layers containing 24 filters of size 5 by 5; the final layer is fully connected; only the first layer has biases. There is no pooling.

The network is overparametrized: it has $154,464$ parameters (compared to $50,000$ training examples).

L.1.2 Network with 34 filters

The model is the same 3-layer convolutional ReLU network as in section L.1.1 except it had 34 units.

The network was still overparametrized: it has $165,784$ parameters (compared to $50,000$ training examples).

L.2 Five layer network

The model is a 5-layer convolutional ReLU network with (with no pooling). It has in the five layers $32, 64, 64, 128$ filters of size 3 by 3; the final layer is fully connected; batch-normalization is used during training.

The network is overparametrized with about $188,810$ parameters (compared to $50,000$ training examples).

L.3 ResNet

The model for the experiments with ResNets was the 56-layer ResNet as detailed in He et al. (2015).

References

M. Anthony and P. Bartlett. *Neural Network Learning - Theoretical Foundations*. Cambridge University Press, 2002.

Andrzej Banburski, Qianli Liao, Brando Miranda, Lorenzo Rosasco, Bob Liang, Jack Hidary, and Tomaso Poggio. Theory III: Dynamics and Generalization in Deep Networks. *arXiv e-prints*, art. arXiv:1903.04991, Mar 2019.

P. Bartlett and J. Shawe-Taylor. Generalization performance of support vector machine and other patern classifiers. In ation performance of support vector machine B. Scholkopf, C. Burges and other patern classifiers (eds.), *Advances in Kernel Methods–Support Vector Learning*. MIT press, 1998.

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

Peter L. Bartlett, Michael I. Jordan, and Jon D. McAuliffe. Convexity, classification, and risk bounds. Technical report, 2003.

Peter L. Bartlett, Dylan J. Foster, and Matus Telgarsky. Spectrally-normalized margin bounds for neural networks. *CoRR*, abs/1706.08498, 2017. URL `http://arxiv.org/abs/1706.08498`.

M. Belkin, S. Ma, and S. Mandal. To understand deep learning we need to understand kernel learning. *ArXiv e-prints*, Feb 2018.

O. Bousquet, S. Boucheron, and G. Lugosi. Introduction to statistical learning theory. pp. 169–207, 2003.

Pratik Chaudhari and Stefano Soatto. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. *CoRR*, abs/1710.11029, 2017. URL `http://arxiv.org/abs/1710.11029`.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-SGD: Biasing Gradient Descent Into Wide Valleys. *arXiv:1611.01838 [cs]*, November 2016. URL `http://arxiv.org/abs/1611.01838`. arXiv: 1611.01838.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. *CoRR*, abs/1703.04933, 2017. URL `http://arxiv.org/abs/1703.04933`.

Simon S Du, Wei Hu, and Jason D Lee. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 384–395. Curran Associates, Inc., 2018. URL `http://papers.nips.cc/paper/7321-algorithmic-regularization-in-learning-deep-homogeneous-models-layers-are-automatically-balanced.pdf`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv e-prints*, art. arXiv:1512.03385, Dec 2015.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. *arXiv:1609.04836 [cs, math]*, September 2016. URL `http://arxiv.org/abs/1609.04836`. arXiv: 1609.04836.

Tengyuan Liang, Tomaso Poggio, Alexander Rakhlin, and James Stokes. Fisher-rao metric, geometry, and complexity of neural networks. *CoRR*, abs/1711.01530, 2017. URL `http://arxiv.org/abs/1711.01530`.

Charles H. Martin and Michael W. Mahoney. Traditional and heavy-tailed self regularization in neural network models. *CoRR*, abs/1901.08276, 2019. URL `http://arxiv.org/abs/1901.08276`.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. Exploring generalization in deep learning. *arXiv:1706.08947*, 2017.

T. Poggio and Q. Liao. Theory II: Landscape of the empirical risk in deep learning. *arXiv:1703.09833, CBMM Memo No. 066*, 2017.

T. Poggio, Q. Liao, B. Miranda, L. Rosasco, X. Boix, J. Hidary, and H. Mhaskar. Theory of deep learning III: explaining the non-overfitting puzzle. *arXiv:1703.09833, CBMM Memo No. 073*, 2017.

T. Poggio, Q. Liao, B. Miranda, A. Banburski, X. Boix, and J. Hidary. Theory IIIb: Generalization in deep networks. *arXiv:1703.09833, CBMM Memo No. 090*, 2018.

Mor Shpigel Nacson, Nathan Srebro, and Daniel Soudry. Stochastic Gradient Descent on Separable Data: Exact Convergence with a Fixed Learning Rate. *arXiv e-prints*, art. arXiv:1806.01796, Jun 2018.

D. Soudry, E. Hoffer, and N. Srebro. The Implicit Bias of Gradient Descent on Separable Data. *ArXiv e-prints*, October 2017.

C. Zhang, Q. Liao, A. Rakhlin, K. Sridharan, B. Miranda, N.Golowich, and T. Poggio. Theory of deep learning IIb: Optimization properties of SGD. *CBMM Memo 072*, 2017.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *CoRR*, abs/1611.03530, 2016. URL `http://arxiv.org/abs/1611.03530`.