CENTER FOR
Brains
Minds+
Machines

CBMM Memo No. 91 **July 11, 2018**

Classical generalization bounds are surprisingly tight for Deep Networks

Qianli Liao[1], **Brando Miranda**[1], **Jack Hidary**[2] **and Tomaso Poggio**[1]
[1]Center for Brains, Minds, and Machines, MIT
[2]Alphabet (Google) X

Abstract

Deep networks are usually trained and tested in a regime in which the training classification error is not a good predictor of the test error. Thus their working regime seems far from generalization, defined as convergence of the empirical to the expected error. Here we show that, when normalized appropriately, deep networks trained on exponential type losses show an approximatevely linear dependence of test loss on training loss. The observation, motivated by a previous theoretical analysis of overparametrization and overfitting, not only demonstrates the validity of classical generalization bounds for deep lerning but suggests that they are tight, directly contradicting the claims of a recent, much cited paper titled “Understanding deep learning requires rethinking generalization”

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

A Surprising Linear Relationship Predicts Test Performance in Deep Networks

Qianli Liao[1], Brando Miranda[1], Jack Hidary[2], and Tomaso Poggio [*2]

[1]Center for Brains, Minds and Machines, MIT
[2]Alphabet (Google) X

July 11, 2018

Abstract

Given two networks with the same training loss on a dataset, when would they have drastically different test losses and errors? Better understanding of this question of generalization may improve practical applications of deep networks. In this paper, we make a step towards answering this question. We show empirically that with the commonly used cross entropy loss for classification, it is surprisingly simple to induce significantly different generalization performances for two networks that have the same architecture and meta parameters: one can either pretrain the networks with different levels of "corrupted" data or simply initialize the networks with weights of different gaussian standard deviations. Further theoretical analysis that follows from a recent theoretical analysis of overfitting in deep networks, shows that these effects are due to an intrinsic problem of measuring test performance with cross entropy training loss. Cross entropy loss can be decomposed in two components that are both minimized by SGD — one of which is not related to expected classification performance. However, if we factor out this irrelevant component of the loss, a surprising linear relationship emerges between training and test losses. With this form of "normalized loss", training loss becomes an excellent predictor of test loss.

[*]To whom correspondence should be addressed

Contents

1 Introduction **3**

2 Observation: Networks that Train Equally but Generalize Differently **3**

3 Theory: Cross Entropy Loss Is Misleading **5**
3.1 Intuition 6
3.1.1 Shallow Linear Network 6
3.1.2 Deep ReLU Network 6
3.2 Comparing empirical minimizers of deep networks 6

4 Experiments: Normalization Leads to Surprising Linear Relationship Between Training and Test Losses **7**
4.1 Interesting observations 7
4.2 Randomly labeled training data 8

5 Discussion **17**

A Results on CIFAR-100 **19**

1 Introduction

Despite many successes of deep networks and a growing amount of research, there remains several fundamental questions unanswered, among which a key puzzle about generalization in deep networks. When does a network generalize well? What is the relationship between training and test performances?

In this paper, we investigate the question of when and why could two deep networks with the same training loss have different testing performances. This question is valuable because training loss is one of the most important clue that deep learning practitioners rely on when making choices of models. It is worth studying how much training loss can tell us about generalization.

In addition to training loss, there are many factors (such as choices of network architecture) that can affect generalization performance and we cannot exhaustively study them in this paper. Therefore, we restrict our models to have the same architecture and training settings within each experiment. We tried different architectures in different experiments and observed consistent results.

2 Observation: Networks that Train Equally but Generalize Differently

First we start with an observation even when two networks have the same architecture, same optimization meta parameters and same resultant training loss, it is possible for them to have different test performances (i.e., error and loss).

We propose two approaches to achieve this effect:

- Initialize networks with different levels of "random pretraining": the network is pretrained on "corrupted" training data for a specified number of epochs (30 in our experiments) — 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: as a commonly used initialization approach, we initialize the weights with diagonal gaussian distribution. All dimension have the same standard deviation. We can simply adjust this standard deviation to get different levels of generalization performance.

We show the results of 'random pretraining" with networks on CIFAR-10 (Figure 1) and CIFAR-100 (Figure 12) and initialization with different standard deviations on CIFAR-10 (Figure 2) and CIFAR-100 (Figure 13).

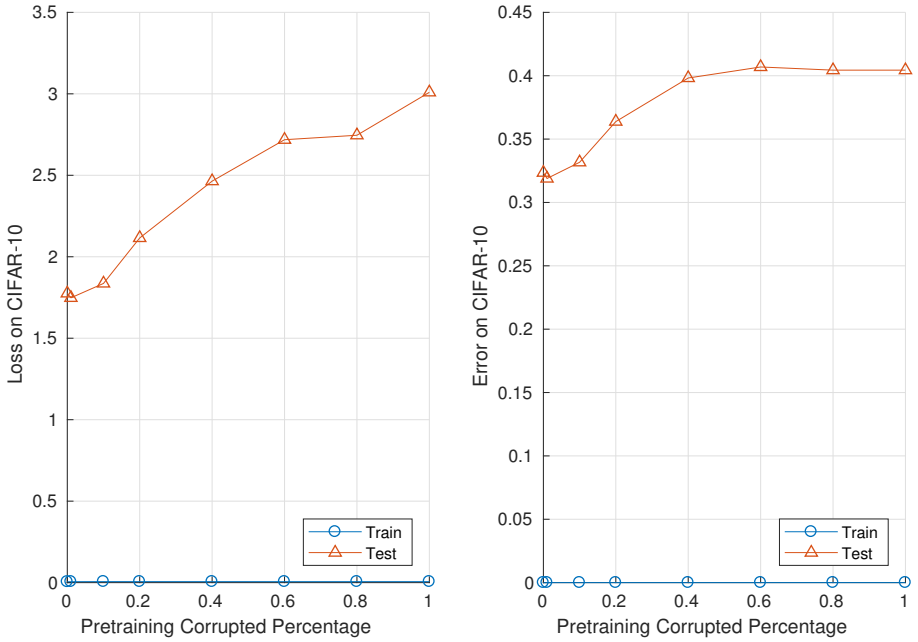

Figure 1: *Random Pretraining vs. Generalization Performance on CIFAR-10: a 5-layer ConvNet (with only convolutional layers and no pooling) is pretrained on training data with partially "corrupted" labels for 30 epochs. Then they are trained on normal data for 80 epochs. Then among the network snapshots saved from all the epochs, we pick a network that is closest to a arbitrary (but low enough) training loss (0.006 here). This is to make sure all models shown in the figure have very close training losses. The number on the x axis indicates what percentage of labels are swapped randomly. As pretraining data gets increasingly "corrupted", the generalization performance of the resultant model gets worse, even though they have similar training losses. Batch normalization (BN) is used. After training, we "absorbed" the means and standard deviations of BN into the network's weights and biases. No data augmentation is performed.*

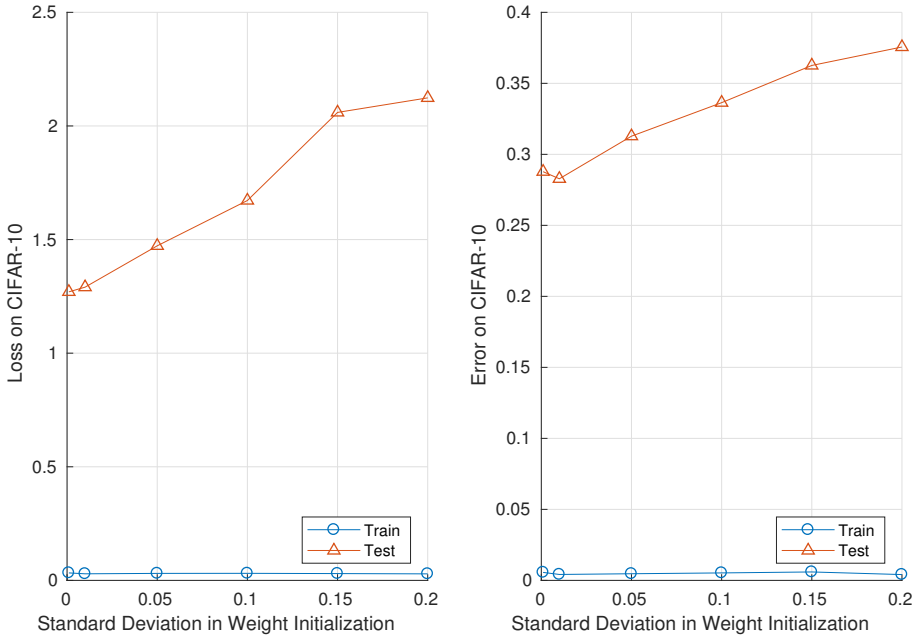

Figure 2: *Standard Deviation in Weight Initialization vs. Generalization Performance on CIFAR-10: the network is initialized with weights of different standard deviations. Other settings are the same as 1. As initial weights get increasingly large norm, the generalization performance of the resultant model gets worse, even though all models have similar training losses.*

3 Theory: Cross Entropy Loss Is Misleading

In the previous section, we observe that it is possible to obtain networks that have the same training loss but very different test performances. This indicates that training loss is not very correlated with test loss. In general, it is quite common to get zero training error (and a very small training loss) when using overparametrized networks. Therefore, comparing models based on training statistics becomes very difficult. In an extreme case, recent work [1] shows that an overparametrized network can fit any random labels and fail to generalize at all. In some sense, training loss seems easily "fooled" by SGD in a way that extremely good training loss can be achieved without any generalization guarantees.

In this section, we theoretically investigate in the case of cross entropy loss why this "super-fitting" phenomenon happens and demonstrate that better fitting does not necessarily lead to better generalization.

3.1 Intuition

3.1.1 Shallow Linear Network

Consider a shallow linear network, trained once with GD on a set of N data x_i, y_i and separately on a different set of N data. Assume that in both cases the problem is linearly separable and thus zero error is achieved on the training set. The question is which of the two solutions will generalize better. The natural approach is to look at the L_2 norm of the solutions: $\sum_n (y_n - w^T x_n)^2$

$$af(w^a, x) = (w^a, x^a) \qquad f(w^b, x^b) = (w^b, x^b) \tag{1}$$

In both cases GD converges to zero loss wih the minimum norm, maximum margin solution for w. Thus the solution with the larger margin (and the smaller norm) should have a lower expected error. In fact, generalization bounds ([2]) appropriate for this case depend on the product of the norm $|w|$ and of a bound on the norm of the data.

3.1.2 Deep ReLU Network

With ReLU nonlinearity, there is a

3.2 Comparing empirical minimizers of deep networks

Notation: We define (as in [3]) 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 $f(W; x) = \sigma(W^K \sigma(W^{K-1} \cdots \sigma(W^1 x)))$, 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 $W^K \in \mathbf{R}^{1, K_l}$. 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.

Consider different zero minima of the empirical risk obtained with the same network on the same training set. Can we predict their expected error from empirical properties only? A natural way to approach the problem of ranking two different minimizers of the empirical risk starts with the "positive homogeneity" property of ReLU networks (see [3]) :

$$f(W^1, \cdots, W_k; x) = \rho_1, \cdots, \rho_K f(\tilde{W^1}, \cdots, \tilde{W_k}; x) \tag{2}$$

where $W_k = \rho_k \tilde{W_k}$ and $||\tilde{W_k}|| = 1$.

This property is valid for layerwise normalization under any norm. Note that $f(W^1, \cdots, W_k; x)$ and $f(\tilde{W^1}, \cdots, \tilde{W_k}; x)$ have the same classification performance on any given (test) set. It follows that different empirical minimizers should be compared in terms of their normalized form: the ρ factors affect exponential type losses loss – driving it to zero by increasing the ρs to infinity – but do not change the classification performance which only depends on the sign of $y_n f(x_n)$. The cross entropy for two classes is given by

$$L = \sum_{n=1}^{N} \ln(1 + e^{-y_n f(x_n)}) = \sum_{n=1}^{N} \ln(1 + e^{-y_n(\rho_1,\cdots,\rho_K) f(\tilde{W}^1,\cdots,\tilde{W}_k;x_n)}). \quad (3)$$

The question is then: what is the right norm to be used to normalize the capacity of deep networks? The "right" normalization should make different normalized minimizers equivalent from the point of view of their intrinsic capacity – for instance equivalent in terms of Radamacher complexity. The positive homogeneity property implies that the correct norm for a multilayer network should be based on a product of norms. Since different norms are equivalent in $\mathbb{R}^n$ it is enough to use L_2.

The argument can be seen more directly looking at a typical generalization bound. It has the following form [4]:

With probability $\geq (1 - \delta)$ $\forall f$

$$|\mathbf{E}(f) - \mathbf{E}_S(f)| \leq 2\mathbb{R}_N(\mathbf{F}) + \sqrt{\frac{\ln(\frac{1}{\delta})}{2N}} \quad (4)$$

where $\mathbf{E}(f)$ is the expected loss, $\mathbf{E}_S(f)$ is the empirical loss, N is the size of the training set and $\mathscr{R}_N(\mathbf{F})$ is the empirical Rademacher complexity of the class of functions $\mathbf{F}$ on the unit sphere to which f belongs.

We expect layerwise normalization to yield the same complexity for the different minimizers. The case of linear functions is a simple example. The Radamacher complexity of a set $\mathbf{F}$) of linear functions $f(x) = w^T x$ can be bounded as $\mathscr{R}_N(\mathbf{F}) \leq \frac{1}{\sqrt{N}} XW$ where X is an L_p bound on the vectors x and W is an L_q bound on the vectors w with L_p and L_q being dual norms, that is $\frac{1}{p} + \frac{1}{q} = 1$. This includes L_2 for both w and x but also L_1 and L_∞. Since different L_p norms are equivalent in finite dimensional spaces (in fact $||x||_p \geq ||x||_q$ when $p \leq q$, with opposite relations holding with appropriate constants) Equation 4 holds under normalization with different L_p norms for w (in the case of linear networks). Notice that the other term in the bound (the last term on the right-hand side of Equation 4) is the same for all networks trained on the same dataset.

The results of [5] – which was the original motivation behind the arguments above and the experiments below– imply that the weight matrices at each layer converge to the minimum Frobenius norm for each minimizer.

4 Experiments: Normalization Leads to Surprising Linear Relationship Between Training and Test Losses

4.1 Interesting observations

- *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 or standard initialization. In fact on the left of figure 3 and on figure

6 one can observe a linear relation for minimizers obtained with default initialization in Pytorch [6] which seems to be a variation of He initialization (check) [7].

- *Robustness* Figures 3, 4, 10, 9, 7, 8 and CITATIONS qianli's layer nets show the linear relationship for different types networks (3 layers vs 5 layers, with and without batch normalization) on CIFAR10. Figures 14 and 15 show the linear relationship on CIFAR100.
- *Norm independence* Figures 4 show that the L_p norm used for normalization does not matter – as expected.

4.2 Randomly labeled training data

Since the Radamacher empirical complexity in Equation 4 does not depend on the labels of the training set, Equation 4 also holds in predicting expected error when the training is on randomly labeled data ([8]) (though different stricter bounds may hold separately for each of the two cases). The experiments in figure 10, 7 and 14 show the expected result. It is quite surprising to see that the new datapoint (corresponding to the randomly trained network) still satisfies aproximatevely (but not exactly) the same linear relationship on figure 10.

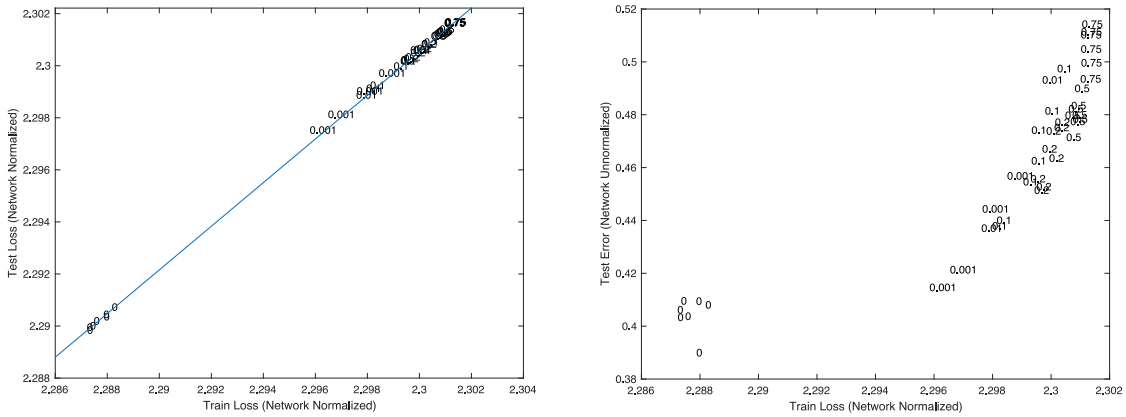

Figure 3: *Test loss/error v.s. training loss with all networks normalized by the Frobenius norm of weights per layer: The model is a 3 layer convolutional ReLU network with the first 2 layers having 24 filters of size 5 by 5; the final layer is fully connected; only the first layer has biases. The network is overparametrized with 154,464 parameters and was trained with 50K examples on CIFAR10. The models were obtained by pre-training on random labels and then by 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. The numbers as markers indicate the amount of corruption of the random labels used in pre-training.*

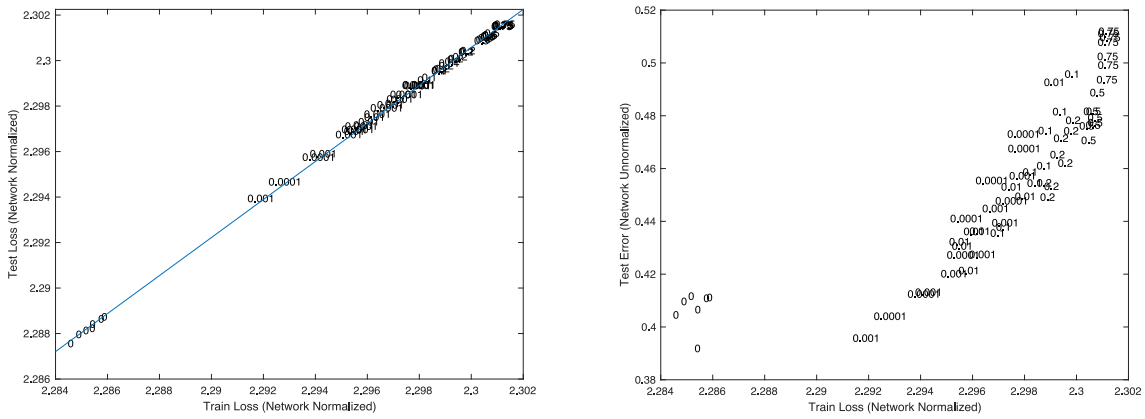

Figure 4: *Test loss/error v.s. training loss with all networks normalized by the L1 norm per layer divided by* 100*: The division by* 100 *is an adjustment to avoid numerical issues because L1 norms are large. The training and the networks were the same as the ones on figure 3. The networks chosen for normalization were the ones at epoch* 300 *because figure 5 shows that the final train loss does not have any effect after 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 as markers indicate the amount of corruption the random labels had during the pre-training.*

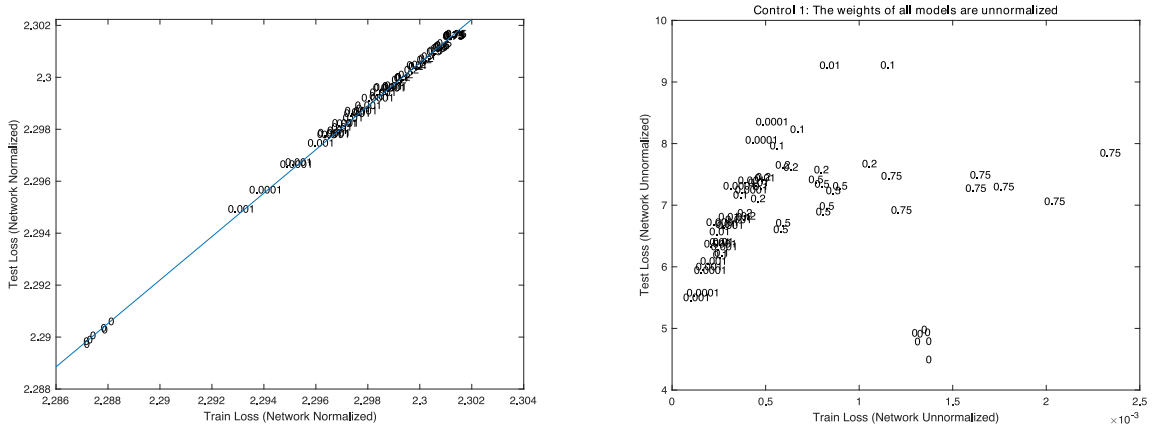

Figure 5: *Left: test loss v.s. training loss with all networks normalized by the Frobenius norm per layer. Right: test loss v.s. training loss with all unnormalized networks. The training and the networks were the same as the ones on figure 3. All networks were trained for* 300 *epochs. Unlike in Figure 3 where the networks had approximately the same loss, in this experiments we made sure the networks converged, had zero training error and did not have the same loss, by choosing the ones at the the end of training (epoch 300). The losses range approximately from* 1.5×10^{-4} *to* 2.5×10^{-3}*. This experiment shows independence of the property of normalize networks from the initial training loss. The numbers as markers indicate the amount of corruption of random labels used during pre-training. 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) was* 4.7651×10^{-5}*.*

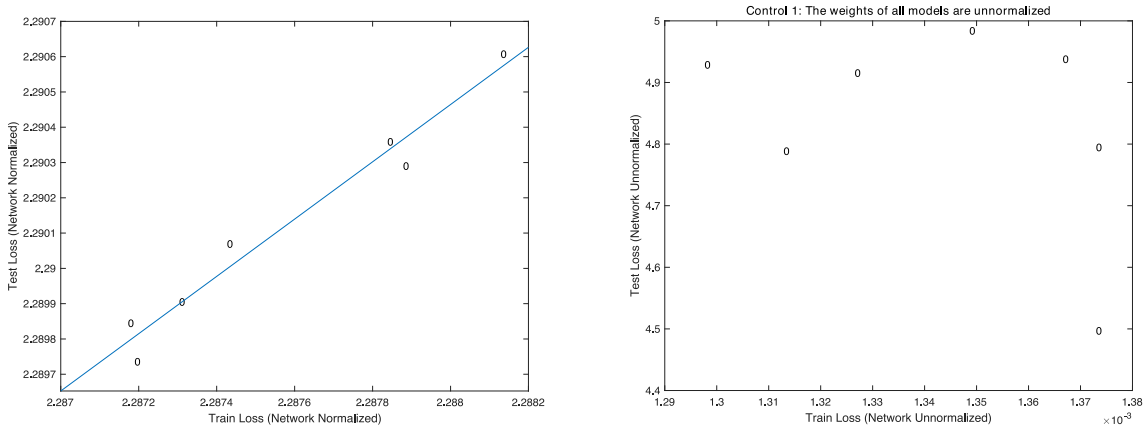

Figure 6: *Left: test loss v.s. training loss with all networks normalized by the Frobenius norm per layer. Right: test loss v.s. training loss with all unnormalized networks. The training and the networks were the same as the ones in figure 3. Since the same conditions hold as in figure 5, the model that was chosen was at epoch* 300*. The losses range approximately from* 1.29×10^{-3} *to* 1.38×10^{-3}*. The 0 as markers indicate there was no corruption during training. The slope and intercept of the line of best fit are* 0.8117 *and* 0.4333 *respectively. The ordinary* R^2 *is* 0.9660 *and the adjusted* R^2 *value is* 0.9592 *while the root mean square (RMSE) was* 6.3624×10^{-5}*.*

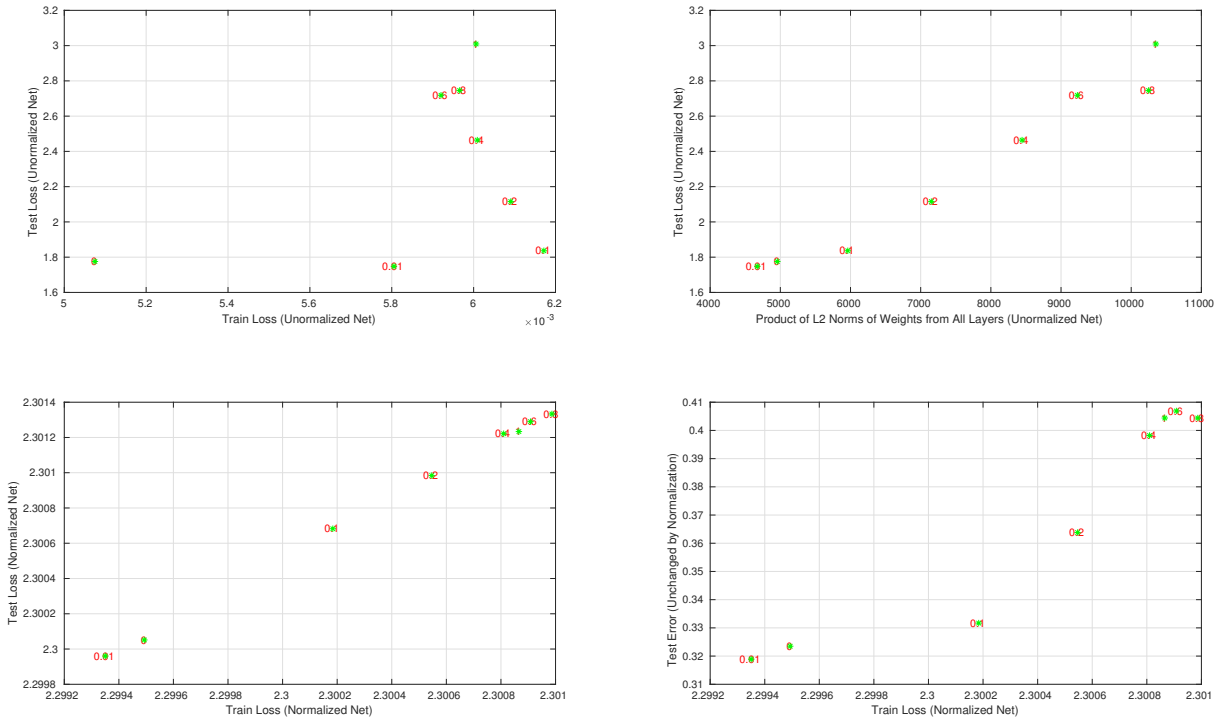

Figure 7: *Random Pretraining experiment with/without normalization on CIFAR-10. The red numbers in the figures indicate the percentages of "corrupted labels" used in pretraining. The green stars (*) in the figures indicate the precise locations of the points.* ***Top left:*** *The training and test losses of unormalized networks. There is no apparent relationship between training and test losses.* ***Top right:*** *the product of L2 norms from all layers of the network. We observe an positive correlation between the norm of the weights and testing loss .* ***Bottom:*** *If we normalize each layer's weights by dividing by its L2 norm, the classification error does not change (bottom right) while the loss do change (bottom left). There is a surprising linear relationship between training and testing losses.*

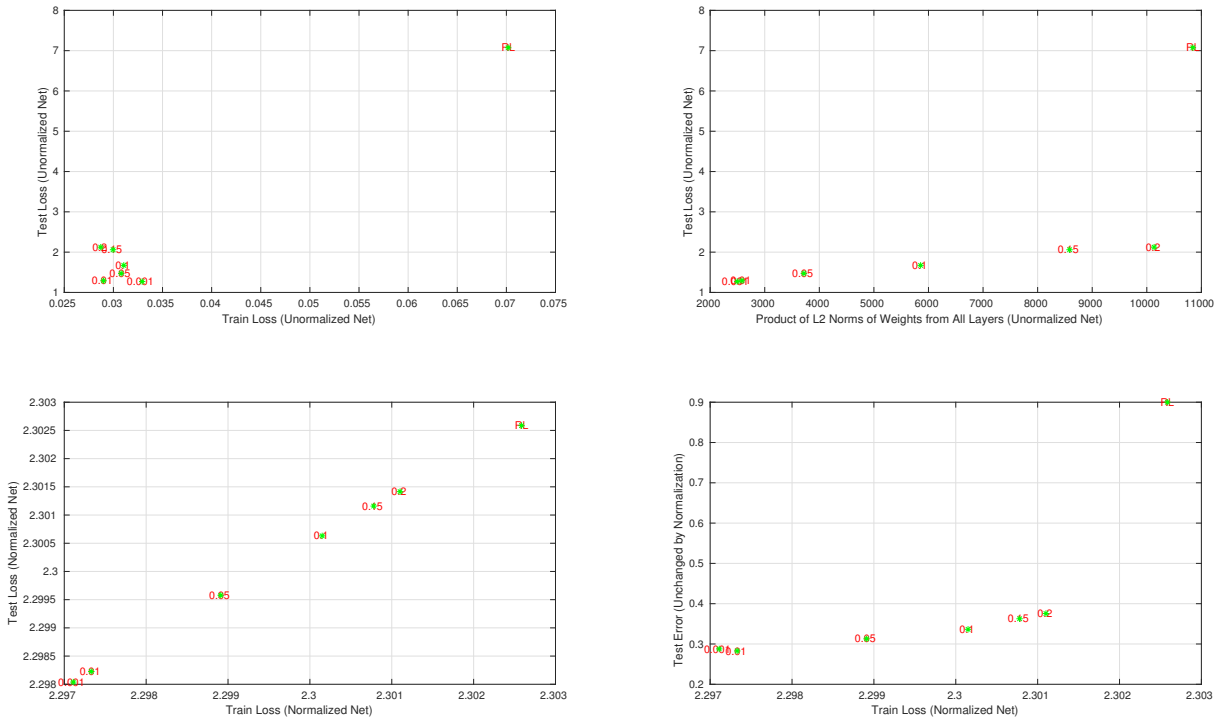

Figure 8: *Same as Figure 7 but not conducted by "random pretraining". Instead, the networks are initialized with different standard deviations (in weights). The red numbers in the figures indicate the standard deviations used in initializing weights. The "RL" point is a reference point that is initialized with standard deviation 0.05 and were trained and tested on completely random labels.*

Note that two unnormalized minimizers of the exponential loss that achieve a given small loss $L = \epsilon$, the minimizer with higher product of the norms $\rho_1, \cdots, \rho_K$ has the higher capacity and thus the highest expected error. Experiments support this claim (see Figure 9 and top left panels of Figure 7), 14, 8, 15.

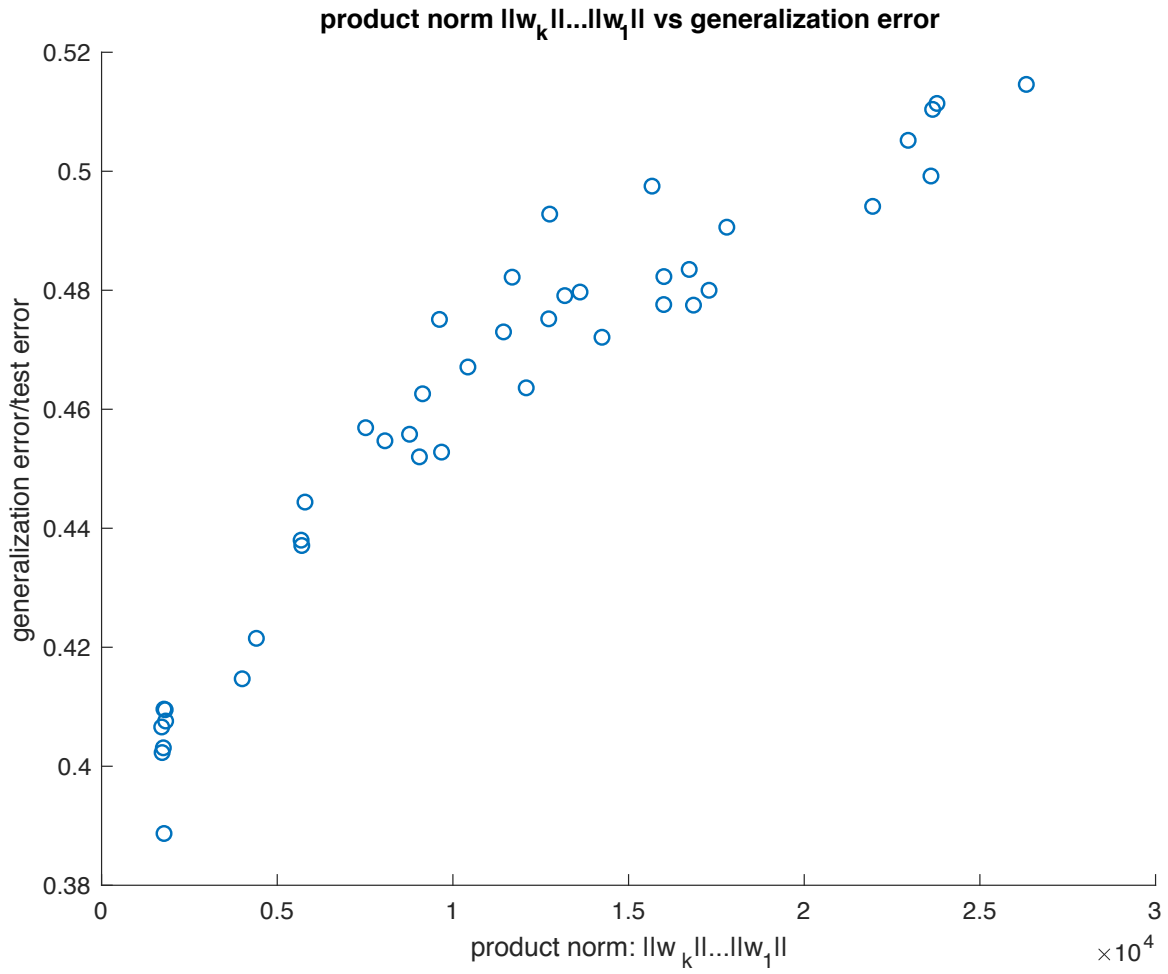

Figure 9: *Plot of test error vs the product norm* $\|W\|_{product} = \prod_{l=1}^{L} \|W_l\|$. *The model here is a 3 layer convolutional ReLU network with the first 2 layers having 24 filters of size 5 by 5; the final layer is fully connected; only the first layer has biases. The network is overparametrized: it has 154,464 parameters and was trained with 50K examples on CIFAR10. The models were obtained by pre-training 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.*

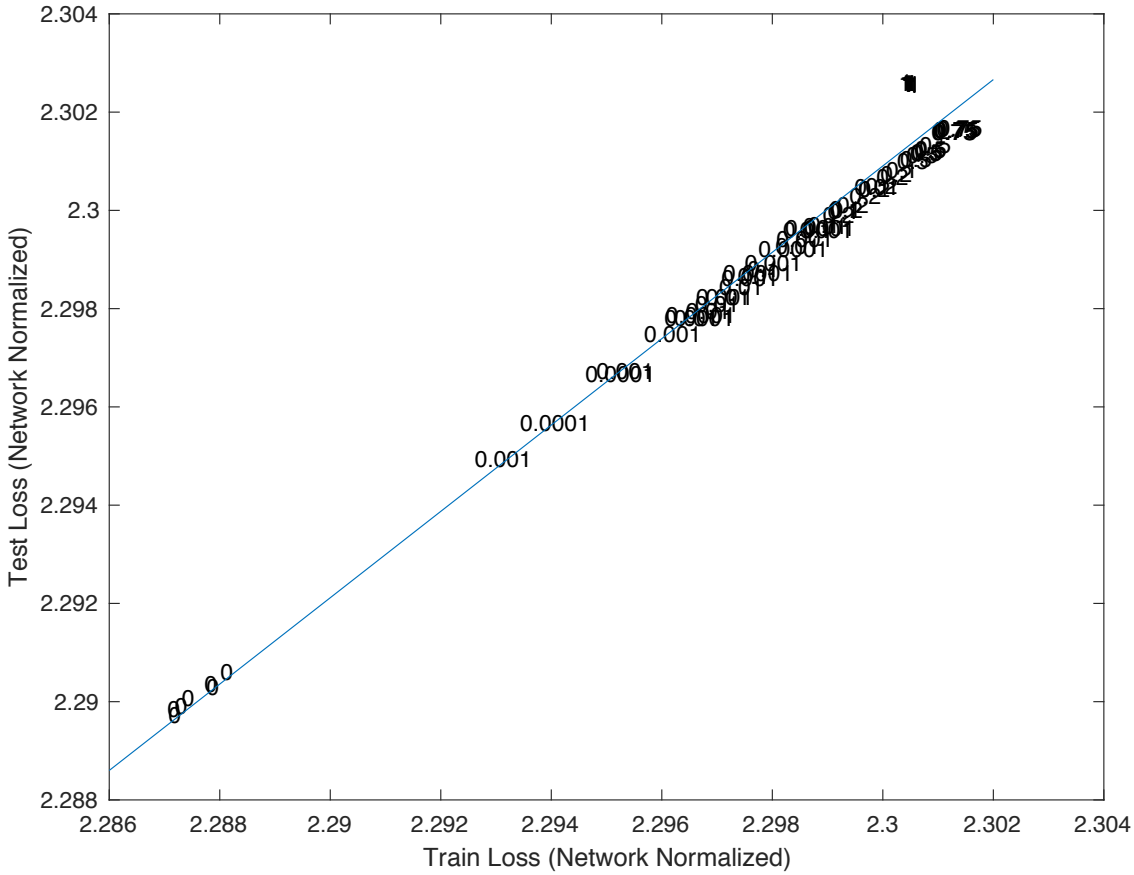

Figure 10: *Plot shows cross entropy loss on the test set v.s. the training loss with all networks normalized by the Frobenius norm per layer. The training and the networks were the same as the ones on figure 3. All networks were trained for* 300 *epochs. Unlike in Figure 3 where the networks had approximately the same loss, in this experiments we made sure the networks converged (and had zero train error) but not had the same loss. Thus the networks that were chosen were the ones at epoch 300. The slope and intercept of the line of best fit are* 0.8789 *and* 0.2795 *respectively. The ordinary and adjusted* R^2 *values are both* 0.9721 *while the root mean square (RMSE) was* 5.8304×10^{-4}*. The numbers as markers indicate the amount of corruption the random labels had during the pre-training had except for the points labels with 1 which correspond to a network trained on random labels and evaluated on random labels. The remaining networks were evaluated on natural labels.*

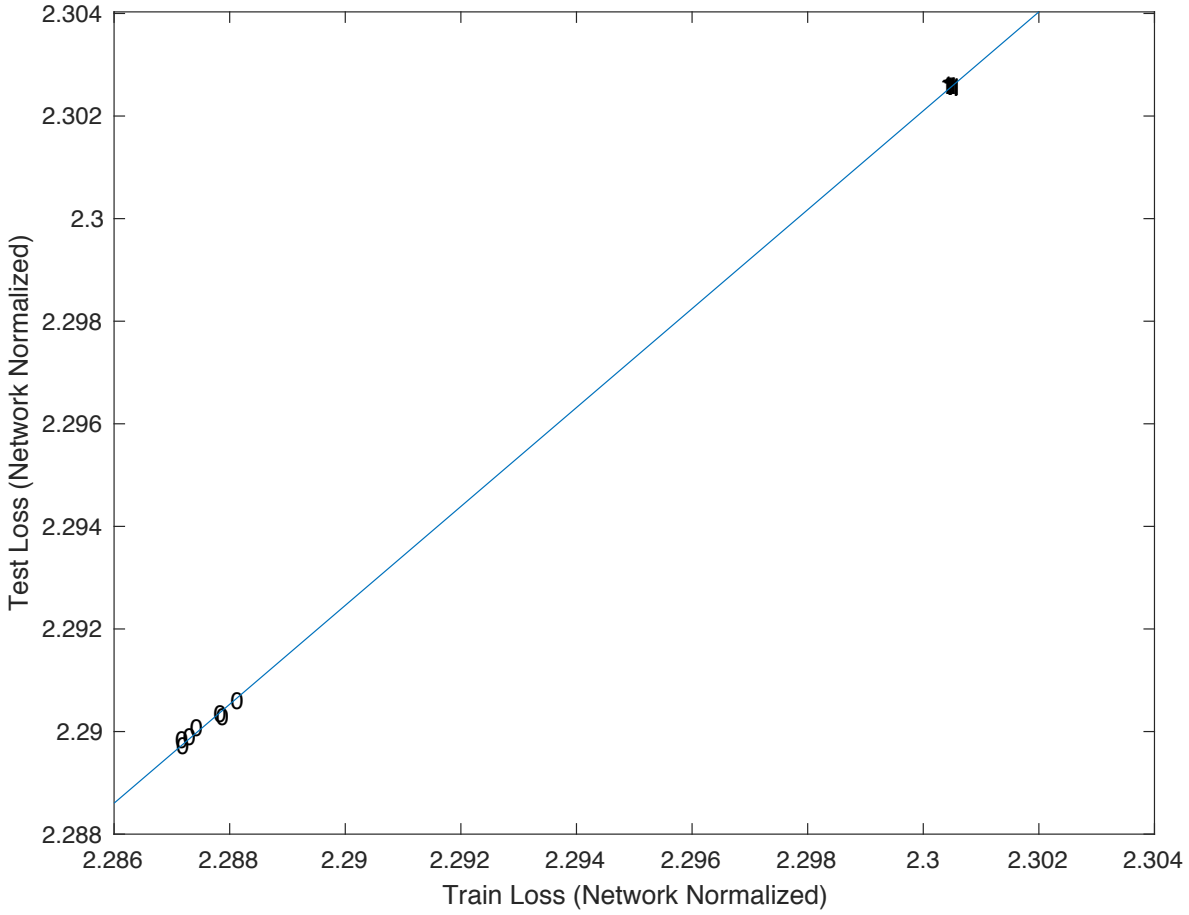

Figure 11: *Plot shows cross entropy loss on the test set v.s. the training loss with all networks normalized by the Frobenius norm per layer. The training and the networks were the same as the ones on figure 3. All networks were trained for* 300 *epochs. Unlike in Figure 3 where the networks had approximately the same loss, in this experiments we made sure the networks converged (and had zero train error) but not had the same loss. Thus the networks that were chosen were the ones at epoch 300. The slope and intercept of the line of best fit are* 0.9642 *and* 0.0844 *respectively. The ordinary and adjusted* R^2 *values are both* 0.9999 *while the root mean square (RMSE) was* 6.9797×10^{-5}*. Similarly as in figure 10, the points labeled 1 were trained only on random labels and evaluated on that same randomly labeled data set. The points marked with 0 were only trained on natural labels.*

5 Discussion

Our results support a measure of complexity in deep networks based on a product norm. The original suggestion [3] is of products of layerwise Frobenius norms but as we discussed earlier, all L_p norms are equivalent in our setup.

The linear relation we found is quite surprising since it implies that the classical bound of Equation 4 is tight *and* holds in a seemingly robust way across different types of networks, different data sets and different initializations. This result not only demonstrates the validity of classical generalization bounds for deep lerning but suggests that they are tight, directly contradicting the claims of a recent, much cited paper titled "Understanding deep learning requires rethinking generalization"

Our results yield a recommendation for practitioners: monitor during training the empirical loss of the normalized network instead of the cross entropy loss (though SGD optimizes the latter). This is the one that matters in terms of stopping time and test performance as suggested by Figures 3 and 4.

More significantly for theory, the observations of this paper clearly demand a critical discussion of some commonly held beliefs such as the key role in generalization of concepts such as dropout, SGD and flat minima.

Acknowledgments

We thank Lorenzo Rosasco, Yuan Yao, Misha Belkin, Xavier Boisch, Andrzej Banbuski and especially Sasha Rakhlin for illuminating discussions. NSF funding provided by CBMM.

References

[1] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[2] Sham M Kakade, Karthik Sridharan, and Ambuj Tewari. On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. In *Advances in neural information processing systems*, pages 793–800, 2009.

[3] 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.

[4] O. Bousquet, S. Boucheron, and G. Lugosi. Introduction to statistical learning theory. pages 169–207, 2003.

[5] 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.

[6] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[8] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *CoRR*, abs/1611.03530, 2016.

APPENDIX

A Results on CIFAR-100

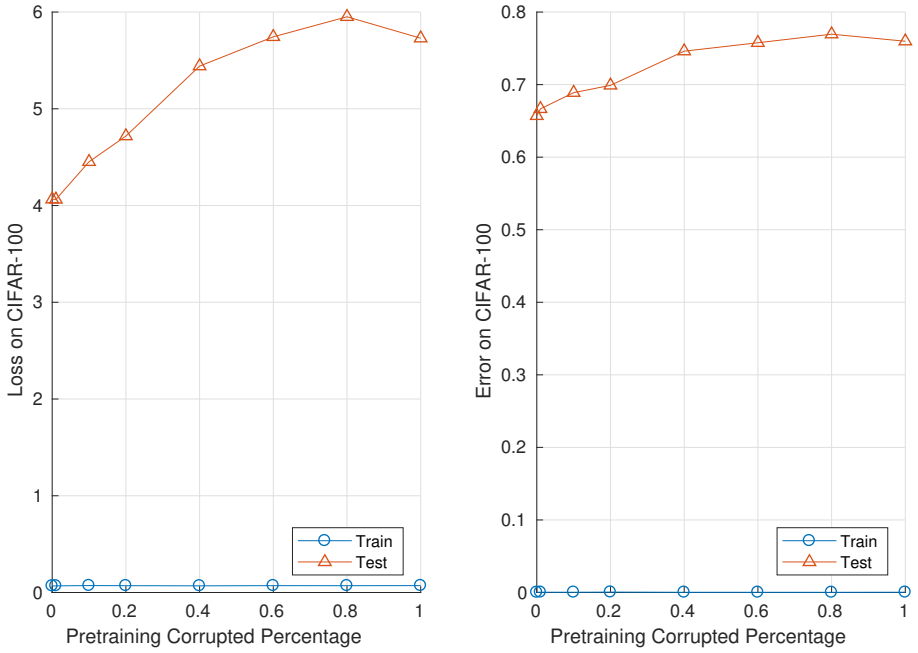

Figure 12: *Same as Figure 1, but on CIFAR-100.*

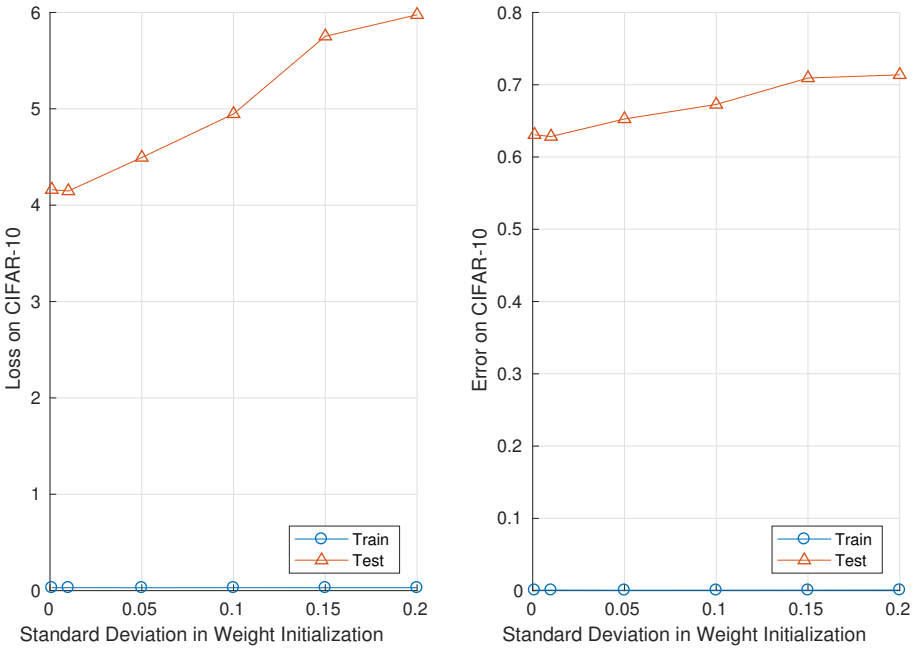

Figure 13: *Same as Figure 2, but on CIFAR-100.*

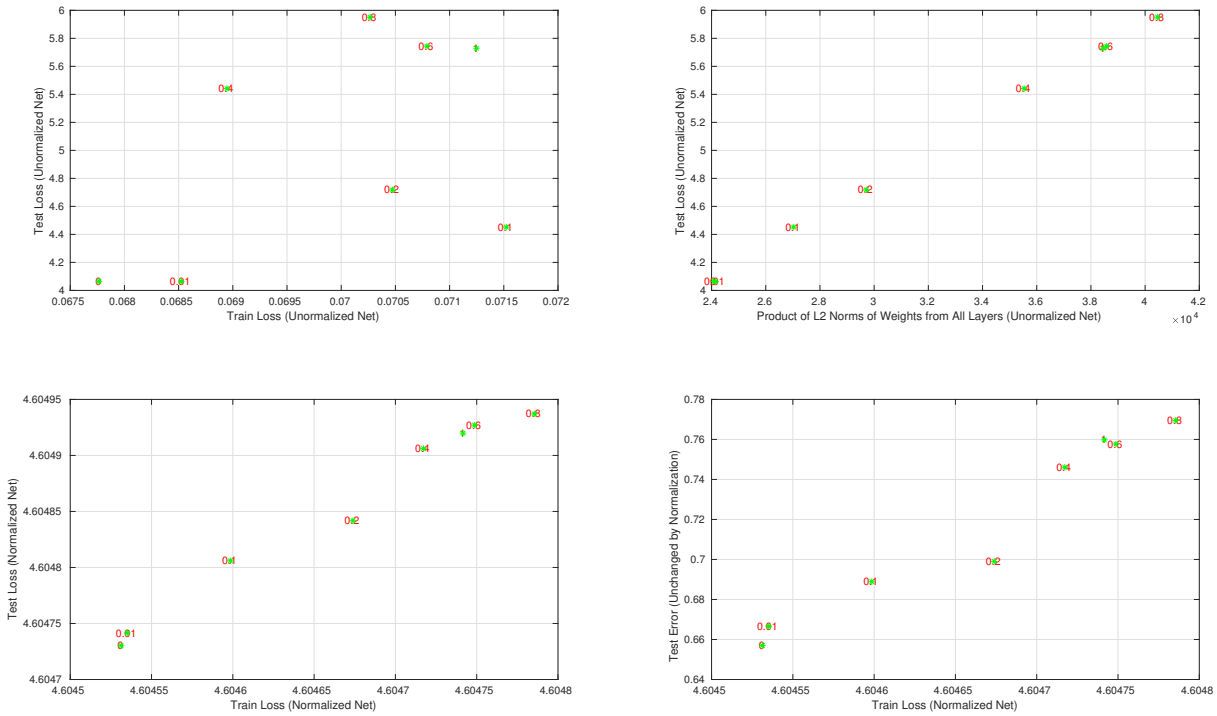

Figure 14: *Same as Figure 7 but on CIFAR-100*

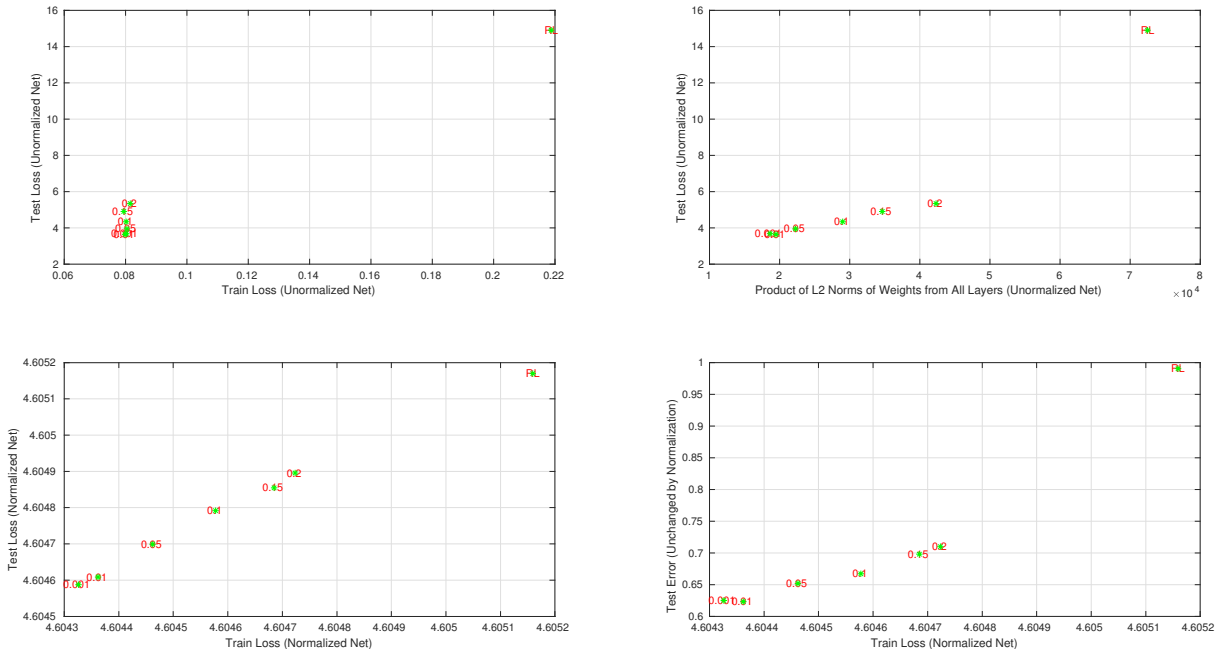

Figure 15: *Same as Figure 8 but on CIFAR-100.*