The intriguing role of module criticality in the generalization of deep networks

Anonymous authors
Paper under double-blind review

Abstract

We study the phenomenon that some modules of deep neural networks (DNNs) are more *critical* than others. Meaning that rewinding their parameter values back to initialization, while keeping other modules fixed at the trained parameters, results in a large drop in the network's performance. Our analysis reveals interesting properties of the loss landscape which leads us to propose a complexity measure, called *module criticality*, based on the shape of the valleys that connects the initial and final values of the module parameters. We formulate how generalization relates to the module criticality, and show that this measure is able to explain the superior generalization performance of some architectures over others, whereas earlier measures fail to do so.

1 Introduction

Neural networks have had tremendous practical impact in various domains such as revolutionizing many tasks in computer vision, speech and natural language processing. However, many aspects of their design and analysis have remained mysterious to this date. One of the most important questions is "what makes an architecture work better than others given a specific task?" Extensive research in this area has led to many potential explanations on why some types of architectures have better performance; however, we lack a unified view that provides a complete and satisfactory answer. In order to attain a unified view on superiority of one architecture over an another in terms of generalization performance, we need to come up with a measure that effectively captures this.

Analyzing the generalization behavior of neural networks has been an active area of research since (Baum & Haussler, 1989). Many generalization bounds and complexity measures have been proposed so far. Bartlett (1998) emphasized the size of the weights in predicting the generalization error. Since then various analysis have been proposed. These results are either based on covering number and Rademacher complexity (Neyshabur et al., 2015; Bartlett et al., 2017; Neyshabur et al., 2019; Long & Sedghi, 2019; Wei & Ma, 2019) or they use approaches similar to PAC-Bayes (McAllester, 1999; Dziugaite & Roy, 2017; Neyshabur et al., 2017; 2018; Arora et al., 2018; Nagarajan & Kolter, 2019a; Zhou et al., 2019). Recently authors have emphasized on the role of distance to initialization rather than norm of the weights in generalization (Dziugaite & Roy, 2017; Nagarajan & Kolter, 2019b; Neyshabur et al., 2019). Earlier results have exponential dependency with depth and focus on fully connected networks. More recently, Long & Sedghi (2019) provided generalization bounds for convolutional neural networks (CNNs) and fully connected networks used in practice and their bounds are size-free and have linear dependency on depth.

Despite the success of earlier works in capturing the dependency of generalization performance of a model with respect to different parameters, they fail to provide a ranking in terms of generalization performance for candidate architectures given a specific task that aligns well with the ground truth. Moreover, majority of these bounds are proposed for fully connected modules and it is not straightforward to evaluate them for different architectures such as ResNets.

Every DNN architecture is a computation graph where each node is a module[1]. We are interested in understanding the role of the fundamental component of a DNN, i.e., module, in generalization

[1]A module is a node in the computation graph that has incoming edges from other modules and outgoing edges to other nodes and performs a linear transformation on its inputs. For a sequential model such as VGG, module definition is equivalent to definition of a layer.

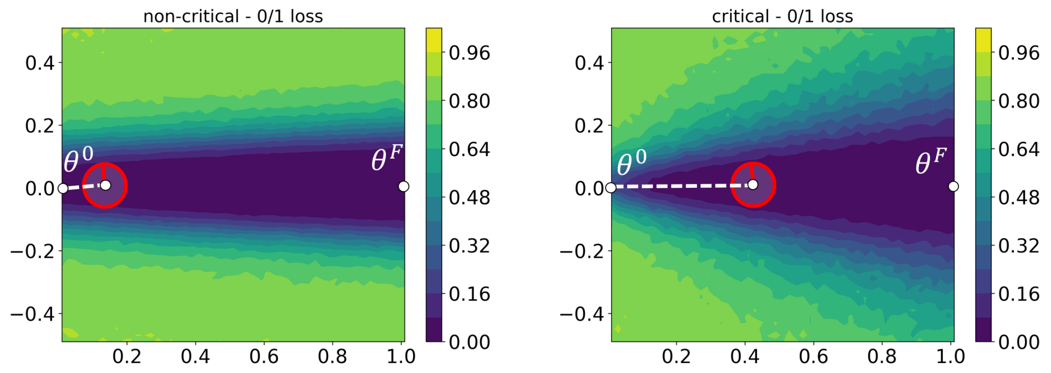


Figure 1: **Module Criticality**: Loss values in the valleys that connect the initial weights θ^0 to the final weights θ^F of a non-critical(left) and a critical(right) module in ResNet18 architecture. Given a ball with radius r (length of the red line), module criticality can be defined as how far one can push the ball in the valley towards initialization (length of the white dashed line) divided by the radius r. Hence, non-critical modules are the ones with a wide valley connecting the initial weight vector to the final one whereas in critical modules, the valley either becomes too sharp or the loss values start to increase when the ball becomes too close to the initial weight. The x axis is simply chosen to be parallel to $\theta^F - \theta^0$ and the y axis is a compact representation of all other dimensions generated by adding Gaussian noise to the the points on the convex combination of θ^0 and θ^F and evaluating the loss. The sign on the y axis is decided based on the sign of the inner product of the noise to θ^0.

and capture how they interact with each other. To do so, we delve deeper into the phenomenon of "module criticality" which was reported by Zhang et al. (2019a). They observed that modules of the network present different robustness characteristics to parameter perturbation. Specifically, if you rewind one module back to its initialization value while keeping the other modules fixed at their trained value, the impact of this perturbation on network performance varies and depends on which module was rewound. Some modules are "critical" meaning that rewinding their value to initialization harms the network performance, while for the others the impact of this perturbation on performance is negligible. They show that various conventional architectures exhibit this dissimilarity phenomenon.

Let us now informally define what we mean by the measure "module criticality" (see Figure 1). For each module, we move on a line from its final trained value to its initialization value (convex combination[2] path) while keeping all other modules the same. Then we measure the size of drop in performance. Let $\theta_i^\alpha = (1-\alpha)\theta_i^0 + \alpha\theta_i^F, \alpha \in [0,1]$ be the convex combination where α_i is the minimum value between 0 and 1 when performance (train error) of the network drops by at most a threshold value ϵ. If α_i is small we can move a long way back to initialization without hurting performance and the "module criticality" of this module would be low. Further, we also wish to incorporate the robustness to noise (that is, the valley width) for the module along this path. If the module is robust to noise along this path (valley is wide) then the module criticality would again be low (see Definition 3.1 for a formal definition).

In this paper, we seek to study this phenomenon in depth and shed some light on by showing that conventional complexity measures cannot capture criticality, see Section 2. Next we theoretically formulate this phenomenon and analyze its role in generalization. Through this path, we provide a new generalization measure that captures the dissimilarity of different modules and point out how it influences the generalization of the corresponding DNN. Intuitively, the closer we can get to initialization for each module, the better the generalization performance.

We analyze the relation between generalization and module criticality through PAC-Bayes analysis. We show that it is not the number of critical modules that matters, but rather the overall network criticality measure as derived in Section 3. If network criticality measure is smaller for an architecture,

[2]A convex combination of two points a linear combination of them where the coefficients are non-negative and sum to 1. Every convex combination of two points lies on the line segment between the points.

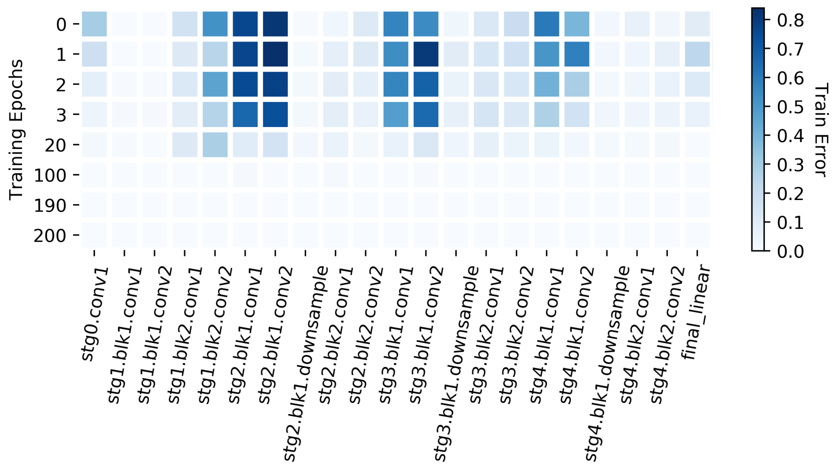


Figure 2: Analysis of rewinding modules to initialization for ResNet-18 architecture. Each row represents a module in ResNet18-v1 that has been rewound to the initialization value and each column represents a particular training epoch. We rewind each module, whereas (Zhang et al., 2019a) rewinds each block.

it has better generalization performance. In Section 4, we demonstrate through various experiments that our proposed measure is able to distinguish between different network architectures in their generalization performance, and the ranking it predicts matches their empirical performance, whereas earlier approaches fail to capture this.

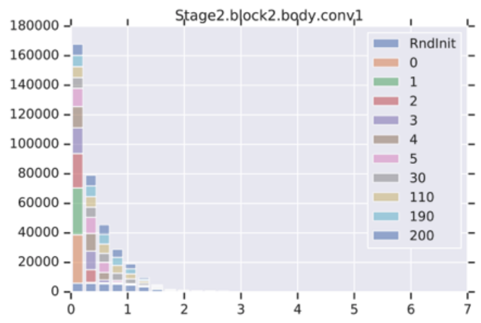
Notation: We use upper case letters for matrices. The operator norm and Frobenius norm of M are denoted by $\|M\|_2$, $\|M\|_{\mathcal{F}}$ respectively. For $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, \ldots, n\}$. Let $\mathcal{L}_S(f)$ be the loss of function f on the training set S with m samples. We are mainly interested in the classification task where $\mathcal{L}_S(f) = \frac{1}{m}\sum_{(x,y)\in S} \mathbf{1}[f(x)[y] \leq \max_{j\neq y} f(x)[j]]$. For any $\gamma > 0$, we also define margin loss $\mathcal{L}_{S,\gamma}(f) = \frac{1}{m}\sum_{(x,y)\in S} \mathbf{1}[f(x)[y] \leq \gamma + \max_{j\neq y} f(x)[j]]$. Let $\mathcal{L}_D(f)$ be the loss of function f on population data distribution D defined similar to $\mathcal{L}_S(f)$. We will denote the function parameterized by Θ by f_Θ.

2 Towards understanding module criticality

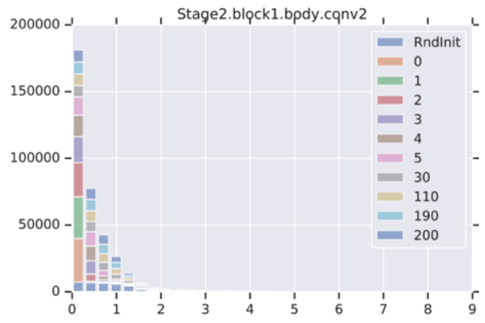
2.1 Setting

A DNN architecture is a directed acyclic computation graph which may or may not be sequential. In order to have a unifying definition between different architectures, we use the notion of "**module**". A module is a node in the computation graph that has incoming edges from other modules and outgoing edges to other nodes and performs a linear transformation on its inputs. For a sequential model such as VGG, module definition is equivalent to definition of a layer. On the other hand, in a ResNet architecture some modules are parallel to each other, e.g., downsample module and the concatenation of two conv modules. Note that, similar to conventional definitions the non-linearity is not part of the module.

Let $\Theta = (\theta_1, \ldots, \theta_d)$ correspond to all parameters of a DNN with d modules, where θ_i refers to the weight matrix (or operator matrix in case of convolution) at module i and θ_i^0, θ_i^F refer to the value of weight matrix at initialization and the end of training, respectively. For sequential architectures, d is equal to the depth of the network but that is not necessarily true for a general architectures such as ResNet.



(a) Ambient module



(b) Critical module

Figure 3: Spectrum of a non-critical and a critical module during different epochs of training.

2.2 Robustness to rewinding

Consider the following perturbation to a trained network at some training epoch as in (Zhang et al., 2019a). For each module in the network, rewind its value back to initialization while keeping the values of all other modules the same. Next, measure the change in performance of the model before and after this manipulation. We repeat a similar analysis that differs from that of (Zhang et al., 2019a) in one detail. Zhang et al. (2019a) rewind the whole resnet block at once, whereas we rewind each module, i.e. each conv in the parallel block, separately. This rewind analysis is shown in Figure 2 for ResNet18. Each column represents a layer in ResNet18-v1 that has been rewound to the initialization value and each row represents a particular training epoch. Similar to earlier analysis, we observe that for many modules of the network, this manipulation does not influence the network performance drastically, while for some others the impact is more noticeable. For example, in Figure 2 we look at effect of rewinding on train error. "Stage2.block1.conv2" module is critical, whereas most other modules, once rewound, do not affect the performance. In Figure 6 in Appendix D we depict the effect of rewinding on different performance criteria (train loss, train error, test error) and note that they exhibit a similar trend.

An stable phenomena: The plots in Figure 2 capture a network trained with SGD with weights initialized with the standard Kaiming initialization (He et al., 2015). To ensure that the observed phenomenon is not an artifact of the training method and the initialization scheme, we repeated the experiments with different initialization and optimization methods. We noted a similar pattern. For example, Figures 7a, 7b in the Appendix illustrate the emerged pattern when changing the initialization to Fixup (Zhang et al., 2019b), replace SGD with Adam (Kingma & Ba, 2014) respectively.

2.3 What measures fail to distinguish critical layers

Spectrum of weight matrices: To understand this phenomenon, we examine how weight matrices evolve during training to see if this can differentiate between critical and non-critical layers. We explore the change in the spectrum of different weight matrices on rewinding and note that the spectrum for a critical and non-critical module look alike. This is shown in Figures 3. We calculate the spectrum using the algorithm of (Sedghi et al., 2019).

Distance to initialization: Next, we analyze the operator norm of difference from initialization for each module. Figure 8 in the appendix depicts this and reveals no difference between critical and non-critical modules. A similar plot was explored in Zhang et al. (2019a), where they find that Frobenius norm and infinity norm also fail to capture criticality.

Change in the activation patterns: We investigate the change in the activation patterns of a network when we rewind a module. To do this we study the similarity between two networks: 1. The original trained network and 2. The network with a rewound module. As a measure of similarity we use CKA (Kornblith et al., 2019). We note that for a non-critical module, the original and rewound networks are similar and in case of a critical module, the similarity between the activation patterns between the two networks breaks down gradually rather than an the exact rewound module. See Figure 9 in Appendix D.

3 CRITICALITY AND GENERALIZATION

Our goal is to understand criticality and how it affects the generalization performance of a DNN. Inspired by the rewind to initialization experiments of (Zhang et al., 2019a), we take one step further and consider changing the value of each module, to the convex combination of its initial and final value, i.e., for each module i, we replace θ_i with $\theta_i^\alpha = (1-\alpha)\theta_i^0 + \alpha\theta_i^F, \alpha \in [0,1]$, and keep all other layers the same. Then we look at the effect of this perturbation on the performance of the network.

Figure 4 depicts how train error, test error and train loss change as we increase the value of α in θ_i^α when i is a critical module (green curve), a non-critical module (orange curve) and all modules (blue curve). We find that along this convex combination path all these performance measures drop monotonically, as we move from the final weights to the initial weights. As the figure suggests, this point is closer to initialization for non-critical modules than the critical ones.

The above experiment shows the effect of moving along a convex combination between module's initial and trained value. To capture the relation between criticality and generalization, we are interested in also accounting for the width of the valley as we move from final value to the initial value. In particular, we are interested in analyzing what happens if we are moving inside a ball of some radius σ_i around each point in this path. PAC-Bayesian analysis, looks for a ball around final value of parameters such that the loss does not change if moving in this ball. Bringing this idea together with the one mentioned above, we are interested in moving from final value to initialization value in a valley of some radius, and find out how much we can move on this path without the ball becoming tighter. Intuitively, the closer we can get to initialization for each module, the better the generalization performance.

Definition 3.1 (Module and Network Criticality). *Given $\epsilon > 0$ and network f_Θ, we define the module criticality for module i as follows:*

$$\mu_{i,\epsilon}(f_\Theta) = \min_{0\leq\alpha,\sigma\leq 1} \left\{ \frac{\alpha_i^2 \left\|\theta_i^F - \theta_i^0\right\|_{\mathcal{F}}^2}{\sigma_i^2} : \mathbb{E}_{u\sim\mathcal{N}(0,\sigma_i^2)}[\mathcal{L}_S(f_{\theta_i^\alpha+u,\Theta_{-i}^F})] \leq \epsilon \right\}, \tag{1}$$

We also define the network criticality as the sum over the modules of the module criticality:

$$\mu_\epsilon(f_\Theta) = \sum_{i=1}^{d} \mu_{i,\epsilon}(f_\Theta) \tag{2}$$

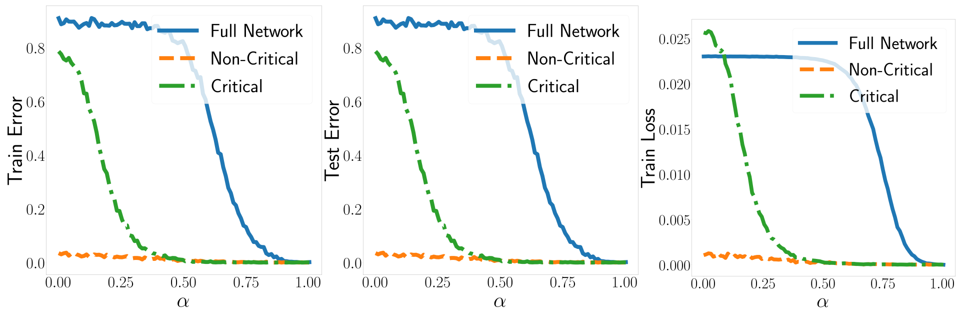
Here, $\mathcal{L}_S$ denotes the empirical zero-one loss over the training set, $f_{\theta_i^\alpha,\Theta_{-i}^F}$ is the DNN's function value where weight matrix corresponding to i^{th} module is replaced by θ_i^α and all other modules are fixed at their values in the end of training, Θ_{-i}^F. $\theta_i^\alpha = (1-\alpha)\theta_i^0 + \alpha\theta_i^F$, where θ_i^0 is the value of the weight matrix at initialization and θ_i^F is the trained value.

Intuitively, network criticality measure is sum of module criticalities. This is also theoretically derived using the analysis below.

3.1 A PAC-BAYESIAN GENERALIZATION BOUND

We attempt to understand the relationship between module criticality, and generalization by deriving a generalization bound using PAC-Bayesian framework (McAllester, 1999). Given a prior distribution over the parameters that is picked in advanced before observing a training set, and a posterior distribution over the parameters that could depend on the training set and a learning algorithm, PAC-Bayesian framework bounds the generalization error in terms of the KullbackLeibler (KL) divergence (Kullback & Leibler, 1951) between the posterior and the prior distribution. We use PAC-Bayesian bounds as they hold for any architecture.

The intuition from Figure 4 suggests moving the parameters of each module as close as possible to initialization before harming the performance. For such α_i, we can then define the posterior Q_i for



(a) Train Error (b) Test Error (c) Train Loss

Figure 4: Performance drop as we move on convex combination path from final to initial value of modules. We find that along this path the training error drops monotonically from the initial weights to the final weights. The blue curve is when we replace all the parameters in the network by the convex combination between their initial and final value simultaneously, the orange curve corresponds to the convex combination path of a single (non-critical) layer and the green curve corresponds to a convex combination path of a critical layer in ResNet-18 architecture.

module i to be a Gaussian distribution centered at θ_i^α with covariance matrix σ_i^2, i.e., as if we have additive noise $u_i \sim \mathcal{N}(0, \sigma_i I)$. We use Θ^α to refer to the case where all θ_i are replaced with $\theta_i^{\alpha_i}$ and matrix U includes all u_i. Then the following theorem holds.

Theorem 3.2. *For any data distribution D, number of samples $m \in \mathbb{N}$, for any $0 < \delta$, for any $0 < \sigma_i \leq 1$ and any $0 \leq \alpha \leq 1$, with probability $1 - \delta$ for the choice of the training set $S_m \sim D$ the following generalization bound holds:*

$$\mathbb{E}_U[\mathcal{L}_D(f_{\Theta^\alpha + U})] \leq \mathbb{E}_U[\mathcal{L}_S(f_{\Theta^\alpha + U})] + \sqrt{\frac{\frac{1}{4}\sum_{i=1}^d k_i \log\left(1 + \frac{\alpha_i^2 \left\|\theta_i^F - \theta_i^0\right\|_{\mathcal{F}}^2}{k_i \sigma_i^2}\right) + \log\frac{m}{\delta} + \tilde{O}(1)}{m-1}}$$

where k_i is the number of parameters in module i. For example, for a convolution module with kernel size $q_i \times q_i$ and number of output channels c_i, $k_i = q_i^2 c_{i-1} c_i$.

The exact bound including the constants and the proof of the above theorem is given in Appendix A. Theorem 3.2 already gives us some insight into generalization of the original network. However, it is not exactly a generalization bound on the original network but rather the perturbed network. We conjecture that for almost any realistic distribution D, any random Θ^0, any Θ^F achieved by known gradient based optimization algorithms, any $0 \leq \alpha \leq 1$ and any $\sigma \geq 0$, the test error does not improve by taking a convex combination of parameters and their initial values followed by Gaussian perturbation. Therefore, we have $\mathcal{L}_D(f_{\Theta^F}) \leq \mathbb{E}_U[L_D(f_{\Theta^\alpha + U})]$. The following corollary restates Theorem 3.2 by using this assumption and doing some search over α and σ parameters in the bound.

Corollary 3.3. *For any data distribution D, number of samples $m \in \mathbb{N}$. For any $\epsilon > 0$, for any $0 < \delta$, if $\mathcal{L}_D(f_{\Theta^F}) \leq \mathbb{E}_U[\mathcal{L}_D(f_{\Theta^\alpha + U})]$ for $u_i \sim \mathcal{N}(0, \sigma_i I)$, then with probability $1 - \delta$ for the choice of the training set $S_m \sim D$, the following generalization bound holds*

$$\mathcal{L}_D(f_\Theta) \leq \epsilon + \sqrt{\frac{\frac{1}{4}\mu'_\epsilon(f_\Theta) + \log\frac{m}{\delta} + \tilde{O}(1)}{m-1}},$$

where $\mu'_\epsilon(f_\Theta)$ is calculated as follows:

$$\mu'_\epsilon(f_\Theta) = \underset{0 \leq \alpha, \sigma \leq 1}{\arg\max}\left\{\sum_i \frac{\alpha_i^2 \left\|\theta_i^F - \theta_i^0\right\|_{\mathcal{F}}^2}{\sigma_i^2} : \mathbb{E}_U[\mathcal{L}_S(f_{\Theta^\alpha + U})] \leq \epsilon\right\}.$$

Note that the above bound uses a slightly different notion of network criticality compare to Definition 3.1 since the bound requires finding α and σ values simultaneously for all modules as opposed to the one defined in Definition 3.1 which allows us to decouple them.

Deterministic Generalization Bound for Convolutional Networks Although PAC-Bayesian bounds are data-dependent hence numerically superior, they provide less insight about the underlying reason that causes generalization. For example, the flatness of the solution after adding Gaussian perturbation can be computed but computing its value does not reveal what properties in the network enforce the loss surface around a point to be flat. Deterministic norm-based generalization bounds on the other hand are numerically much looser yet provide better insights into dependency of generalization to different parameters. In Appendix B, we build on the results of the Theorem 3.2 to present a norm-based deterministic bound using module criticality.

In this section, we intuitively justified network criticality measure and proved the form of dependency of DNN generalization to network criticality measure in Corollary 3.3. In Section 4, we empirically show that network criticality measure is able to correctly rank generalization performance of different architectures while earlier measures fail to do so.

4 Experiments

We perform several experiments to compare our network criticality measure to earlier complexity measures in the literature. All our experiments are performed on CIFAR10 dataset. For all experiments, implementation and architecture details are presented in Appendix C.

Table 1 summaries the quantities that are calculated in this Section. For the last two measures, we calculate σ_i and α_i as in Definition 3.1.

Table 1: Quantities of Interest

Generalization Error (GE)	$\mathcal{L}_D(f_\Theta) - \mathcal{L}_S(f_\Theta)$
Product of Frobenius Norms (PFN)	$\Pi_i \lVert\theta_i^F\rVert_\mathcal{F}$
Product of Spectral Norms (PSN)	$\Pi_i \lVert\theta_i^F\rVert_2$
Distance to Initialization (DtI)	$\sum_i \lVert\theta_i^0 - \theta_i^F\rVert_\mathcal{F}^2$
Number of Parameters (NoP)	Total number of parameters in the network
PAC Bayes (at error threshold 0.1)	$\sum_i \lVert\theta_i^0 - \theta_i^F\rVert_\mathcal{F}^2/\sigma_i^2$
Network Criticality Measure (at error threshold 0.1)	$\sum_i \alpha_i^2 \lVert\theta_i^0 - \theta_i^F\rVert_\mathcal{F}^2/\sigma_i^2$

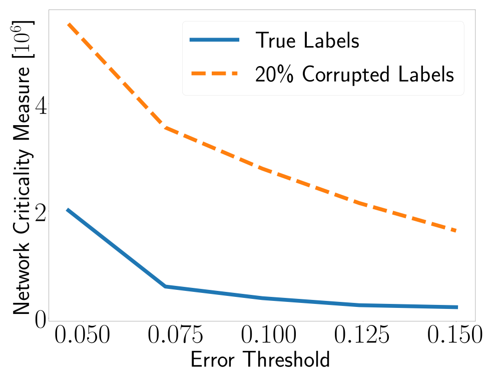
First as a sanity check we use our complexity measure (lower is better) to compare between a ResNet18 trained on true labels and compare it with a ResNet18 trained on data where 20% of the labels are randomly corrupted. As seen in Figure 5a, our measure is able to correctly capture that the network trained with true label generalizes better than the one trained on corrupted labels (4.62% error vs. 35%).

Next, in Table 2 (and in Figure 5b) we compare the generalization performance of four ResNet18 architectures where we vary the number of output channels in each stage. In the ResNet18 (1x width) the number of output channels are 16, 16, 32, 64, 128 in the five stages respectively. The other ResNet18s have output channels scaled by factors (2x,4x and 8x) in each stage. There is a particular ranking of the networks based on their generalization error and it is desired for complexity measures to capture it. Therefore, we compare the rankings proposed by network criticality measure and complexity measures from the literature with the empirical rankings obtained in the experiment. To do this we calculate the *Kendall's τ correlation coefficient* (Kendall, 1938) which is defined as follows:

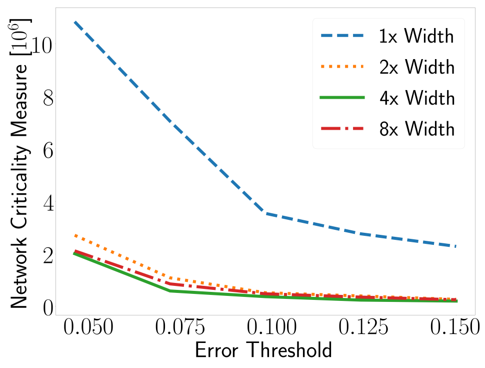
$$\text{Kendall's } \tau = \frac{\#\text{ of pairs where the rankings agree} - \#\text{ of pairs where the rankings disagree}}{\#\text{ pairs}}.$$

This coefficient lies between -1 and 1, where 1 denotes a high correlation between the two set of rankings. Table 2 shows that the Kendall's τ coefficient between our network criticality measure and the generalization error is higher than all other complexity measures that we compared to. Moreover, our complexity measure only fails to correctly capture the relative rank of one pair – ResNet18 (4x width) vs. ResNet18 (8x width).

Finally, we repeat the above experiment on different conventional DNN architectures (see Table 3 for details). We note that network criticality measure correctly predicts the ranking of generalization



(a) Comparing ResNet18 on trained true labels vs. corrupted labels



(b) ResNet18 architectures obtained by scaling up the number of output channels in each stage. The ResNet (1x width) has 16, 16, 32, 64, 128 output channels respectively in its 5 stages.

Figure 5: Network criticality measure as a function of error threshold, for ResNet18.

Table 2: Comparing generalization measures on ResNet18 with different channel widths. The network is ResNet18 with 16,16,32,64,128 channels in the five stages and is scaled by different factors.

ResNet18 (x)	GE	PFN	PSN	DtI	NoP	PAC Bayes	Net. Criticality
1x width	6.27%	4.5e17	2.3e15	1409	6.9e6	6.5e7	3.5e7
2x width	5.13%	8.2e19	6.4e18	2253	2.7e7	2.8e6	8.2e6
4x width	4.61%	7.6e21	9.5e20	3433	1.1e8	9.3e6	6.6e6
8x width	2.88%	1.1e24	3.28e22	5365	4.4e8	9.8e6	7.0e6
Kendall's τ	-	-1	-1	-1	-1	0	0.66

performance for various different architectures – ResNet18, ResNet34, ResNet101, VGG16 and a 3-layer fully connected network (FCN). We find that the generalization error of ResNet34 is the lowest which is correctly captured only by our complexity measure and not by any other measures. Further, the Kendall's τ correlation coefficient between the ranking based of the generalization error and our network criticality measure is 0.8 which is again higher than this coefficient for any other complexity measure. Our measure only fails to capture the relative ranking of ResNet101 and VGG16. The difference in the generalization error of these networks in less than 1% which makes this challenging.

Table 3: Comparing complexity measures on different architectures.

Network	GE	PFN	PSN	DtI	NoP	PAC Bayes	Net. Criticality
ResNet18	4.61%	7.6e21	9.5e20	3433	1.1e8	9.3e6	6.6e6
ResNet34	4.52%	3.0e37	3.1e34	4804	2.1e8	1.2e7	6.3e6
ResNet101	6.4%	8.4e109	2.3e99	18630	4.2e8	3.4e7	3.5e7
VGG16	7.47%	5.1e15	8.6e12	2341	3.3e8	1.1e7	1.3e7
FCN	29.83%	8.6e7	6.2e5	35964	2.0e8	8.9e7	1.0e8
Kendall's τ	-	-0.6	-0.6	0.2	0	0.4	0.8

5 Conclusion

In this paper, we studied the module criticality phenomenon and proposed a complexity measure based on module criticality that is able to correctly predict the superior performance of some DNN architectures over others, for a specific task. We believe module criticality can be used as a road-map for designing new task-specific architectures. Proposing new regularizers that improve generalization performance by bounding criticality or spreading it among various modules of the network is an exciting direction for future work.

References

Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. *arXiv preprint arXiv:1802.05296*, 2018.

Peter L Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE transactions on Information Theory*, 44(2):525–536, 1998.

Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pp. 6240–6249, 2017.

Eric B. Baum and David Haussler. What size net gives valid generalization? In D. S. Touretzky (ed.), *Advances in Neural Information Processing Systems 1*, pp. 81–90. Morgan-Kaufmann, 1989. URL http://papers.nips.cc/paper/154-what-size-net-gives-valid-generalization.pdf.

Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *UAI*, 2017.

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*, pp. 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. *ICML*, 2019.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

John Langford and Rich Caruana. (not) bounding the true error. In *Advances in Neural Information Processing Systems*, pp. 809–816, 2002.

Philip M Long and Hanie Sedghi. Size-free generalization bounds for convolutional neural networks. *arXiv preprint arXiv:1905.12600*, 2019.

David A McAllester. Pac-bayesian model averaging. In *COLT*, volume 99, pp. 164–170. Citeseer, 1999.

Vaishnavh Nagarajan and J Zico Kolter. Deterministic pac-bayesian generalization bounds for deep networks via generalizing noise-resilience. *arXiv preprint arXiv:1905.13344*, 2019a.

Vaishnavh Nagarajan and J Zico Kolter. Generalization in deep networks: The role of distance from initialization. *arXiv preprint arXiv:1901.01672*, 2019b.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pp. 1376–1401, 2015.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pp. 5947–5956, 2017.

Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *ICLR*, 2018.

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. *ICLR*, 2019.

Konstantinos Pitas, Mike Davies, and Pierre Vandergheynst. Pac-bayesian margin bounds for convolutional neural networks. *arXiv preprint arXiv:1801.00171*, 2017.

Hanie Sedghi, Vineet Gupta, and Philip M Long. The singular values of convolutional layers. *ICLR*, 2019.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Colin Wei and Tengyu Ma. Data-dependent sample complexity of deep neural networks via lipschitz augmentation. *arXiv preprint arXiv:1905.03684*, 2019.

Chiyuan Zhang, Samy Bengio, and Yoram Singer. Are all layers created equal? *arXiv preprint arXiv:1902.01996*, 2019a.

Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321*, 2019b.

Wenda Zhou, Victor Veitch, Morgane Austern, Ryan P. Adams, and Peter Orbanz. Non-vacuous generalization bounds at the imagenet scale: a PAC-bayesian compression approach. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJgqqsAct7.

APPENDIX

A PROOF OF THE THEREORM 3.2

We start by stating the PAC-Bayes theorem which bounds the generalization error of any posterior distribution Q on parameters Θ that can be reached using the training set given a prior distribution P on parameters that should be chosen in advance and before observing the training set.

Theorem A.1 ((McAllester, 1999)). *For any data distribution D, number of samples $m \in \mathbb{N}$, training set $S_m \sim D$, and prior distribution P on parameters Θ, posterior distribution Q, for any $0 < \delta$, with probability $1-\delta$ over the draw of training data we have that*

$$KL\left(\mathbb{E}_{\Theta\sim Q}[\mathcal{L}_S(f_\Theta)]\middle|\middle|\mathbb{E}_{\Theta\sim Q}[\mathcal{L}_D(f_\Theta)]\right) \leq \frac{KL(Q||P) + \log\frac{m}{\delta}}{m-1}$$

where KL is the the KullbackLeibler (KL) divergence (Kullback & Leibler, 1951).

Following (Dziugaite & Roy, 2017), we use the inequality $\mathrm{KL}^{-1}(q|c) \leq q + \sqrt{c/2}$ to achieve a simple bound on the test error:

$$\begin{aligned}\mathbb{E}_{\Theta\sim Q}[\mathcal{L}_D(f_\Theta)] &\leq \mathrm{KL}^{-1}\left(E_{\Theta\sim Q}[\mathcal{L}_S(f_\Theta)]\middle|\frac{\mathrm{KL}(Q||P)+\log\frac{m}{\delta}}{m-1}\right)\\ &\leq \mathbb{E}_{\Theta\sim Q}[\mathcal{L}_S(f_\Theta)] + \sqrt{\frac{\mathrm{KL}(Q||P)+\log\frac{m}{\delta}}{2(m-1)}},\end{aligned}$$

where $\mathrm{KL}^{-1}(q|c) = \sup\{p \in [0,1] : \mathrm{KL}(q||p) \leq c\}$.

The intuition from Figure 4 suggests that moving the parameters of each module as close as possible to initialization before harming the performance. For such α_i, we can then define the posterior Q_i for module i to be a Gaussian distribution centered at θ_i^α with covariance matrix σ_i^2, i.e., as if we have additive noise $u_i \sim \mathcal{N}(0, \sigma_i I)$. We use Θ^α to refer to the case where all θ_i are replaced with $\theta_i^{\alpha_i}$ and matrix U includes all u_i. Then the training loss term can be decomposed as

$$\begin{aligned}\mathbb{E}_{\Theta\sim Q}[\mathcal{L}_S(f_\Theta)] &= \mathbb{E}_{u_i\sim\mathcal{N}(0,\sigma_i I)}[\mathcal{L}_S(f_{\Theta^\alpha+U})]\\ &\leq \mathcal{L}_S(f_{\Theta^F}) + \left|\mathbb{E}_{u_i\sim\mathcal{N}(0,\sigma_i I)}[\mathcal{L}_S(f_{\Theta^\alpha+U})] - \mathcal{L}_S(f_{\Theta^F})\right|,\end{aligned}$$

where the second term on the right hand side of the inequality captures the flatness of the point Θ^α by adding Gaussian noise and measuring the change in the loss. Therefore, searching over the posterior corresponds to finding a flat solution in the valley that connects the initial and final points. Next, we use this intuition to prove a generalization bounds based on module criticality.

First, we express the value of the $\tilde{O}(1)$ term in Theorem 3.2 , which is equal to ϵ as below

$$\epsilon = \sum_i \log\left(7m + 2\log\left(\frac{k_i}{k_i\sigma_i^2 + \alpha_i^2\left\|\theta_i^F - \theta_i^0\right\|_{\mathcal{F}}^2}\right)\right). \quad (3)$$

Now we proceed with the proof.

The KL-divergence between two k-dimensional Gaussian distributions can be written as

$$\mathrm{KL}(\mathcal{N}(\mu_1,\Sigma_1)||\mathcal{N}(\mu_P,\Sigma_P)) = \frac{1}{2}\left[\mathrm{tr}\left(\Sigma_2^{-1}\Sigma_1\right) + (\mu_2-\mu_1)^\top\Sigma_2^{-1}(\mu_2-\mu_1) - k + \ln\left(\frac{\det\Sigma_2}{\det\Sigma_1}\right)\right].$$

The above equation can be further simplified for Gaussian distributions with a diagonal covariance matrix. Let the prior P be a Gaussian distribution such that for each module i, the distribution is

$\mathcal{N}(\theta_i^0, \sigma_{P,i}^2 I)$ and let the posterior Q be a Gaussian distribution such that for each module i, the distribution is $\mathcal{N}((1-\alpha)\theta_i^0 + \alpha\theta_i^F), \sigma_{Q,i}^2 I)$. We can then write the KL-divergence $\mathrm{KL}(Q||P)$ as

$$\mathrm{KL}(Q||P) = \frac{1}{2}\sum_i \left[\frac{k_i\sigma_{Q,i}^2 + \alpha_i^2 \left\|\theta_i^F - \theta_i^0\right\|_{\mathcal{F}}^2}{\sigma_{P,i}^2} - k_i + k_i \log\left(\frac{\sigma_{P,i}^2}{\sigma_{Q,i}^2}\right)\right]. \tag{4}$$

Since prior should be decided before observing the training set, we are not allowed to optimize for $\sigma_{P,i}$ directly. However, one can optimize for $\sigma_{P,i}$ over a pre-defined set of values and use a union bound argument to get the generalization bound for the best $\sigma_{P,i}$ in that set. We use a covering approach suggested by Langford & Caruana (2002). For $b, \epsilon > 0$, if one chooses the variance of prior to be $\exp(-\epsilon j + b)$ for $j \in \mathbb{N}$ such that for each j the bound holds with probability$1 - \frac{6}{\pi^2 j^2}$, then all bounds hold with probability $1 - \sum_{j\in\mathbb{N}} \frac{6}{\pi^2 j^2} = 1 - \delta$. We can apply the same idea to every module such that the bound holds with probability $1 - \delta \prod_{i=1}^d \frac{6}{\pi^2 j_i^2}$.

If we choose $\sigma_{Q,i}^2 \leq 1$ then we have $\sigma_{P,i} \leq \exp\left(\frac{4m}{k_i} + 1\right)$. Otherwise, the bound holds since the right hand side is greater than one. Given (from Equation 3) $\epsilon \geq 0$, if we choose $\sigma_{P,i}^2$ to have the form $\exp\left(\frac{4m - j_i}{k_i} + 1\right)$, for some integer j_i, we can always find choose j_i such that

$$k_i\sigma_{Q,i}^2 + \alpha_i^2 \left\|\theta_i^F - \theta_i^0\right\|_{\mathcal{F}}^2 \leq k_i\sigma_{P,i}^2 \leq \exp(1/k_i)\left(k_i\sigma_{Q,i}^2 + \alpha_i^2 \left\|\theta_i^F - \theta_i^0\right\|_{\mathcal{F}}^2\right). \tag{5}$$

Therefore, the KL-divergence can be bounded as

$$\begin{aligned}
\mathrm{KL}(Q||P) &\leq \frac{1}{2}\sum_i \left[k_i \frac{k_i\sigma_{Q,i}^2 + \alpha_i^2 \left\|\theta_i^F - \theta_i^0\right\|_{\mathcal{F}}^2}{k_i\sigma_{Q,i}^2 + \alpha_i^2 \left\|\theta_i^F - \theta_i^0\right\|_{\mathcal{F}}^2} - k_i + k_i \log\left(\frac{\sigma_{P,i}^2}{\sigma_{Q,i}^2}\right)\right] \\
&= \frac{1}{2}\sum_i \left[k_i \log\left(\frac{\sigma_{P,i}^2}{\sigma_{Q,i}^2}\right)\right] \\
&\leq \frac{1}{2}\sum_i k_i \log\left(\frac{\exp(1/k_i)\left(k_i\sigma_{Q,i}^2 + \alpha_i^2 \left\|\theta_i^F - \theta_i^0\right\|_{\mathcal{F}}^2\right)}{k_i\sigma_{Q,i}^2}\right) \\
&\leq \frac{1}{2}\sum_i k_i \log\left(\frac{\exp(1/k_i)\left(k_i\sigma_{Q,i}^2 + \alpha_i^2 \left\|\theta_i^F - \theta_i^0\right\|_{\mathcal{F}}^2\right)}{k_i\sigma_{Q,i}^2}\right) \\
&\leq \frac{1}{2}\sum_i 1 + k_i \log\left(1 + \frac{\alpha_i^2 \left\|\theta_i^F - \theta_i^0\right\|_{\mathcal{F}}^2}{k_i\sigma_{Q,i}^2}\right).
\end{aligned}$$

Note that in order to achieve the inequality in equation 5, j_i should be chosen as

$$j_i = \left\lfloor \frac{4m}{k_i} + 1 + \log\left(\frac{k_i}{k_i\sigma_{Q,i}^2 + \alpha_i^2 \left\|\theta_i^F - \theta_i^0\right\|_{\mathcal{F}}^2}\right)\right\rfloor \leq 5m + \log\left(\frac{k_i}{k_i\sigma_{Q,i}^2 + \alpha_i^2 \left\|\theta_i^F - \theta_i^0\right\|_{\mathcal{F}}^2}\right).$$

Given that each such bound should hold with probability $1 - \delta \prod_{i=1}^d \frac{6}{\pi^2 j_i^2}$, the log term in the bound can be written as

$$\log\frac{m}{\delta} + \sum_i \log(\pi^2 j_i^2/6) \leq \log\frac{m}{\delta} + 2\sum_i \log\left(7m + 2\log\left(\frac{k_i}{k_i\sigma_{Q,i}^2 + \alpha_i^2 \left\|\theta_i^F - \theta_i^0\right\|_{\mathcal{F}}^2}\right)\right).$$

Putting everything together proves the Theorem statement.

B A Deterministic Generalization Bound for Convolutional Networks

We start by stating a generalization bound given in (Neyshabur et al., 2018) with a slight improvement in the constants.

Lemma B.1 (Neyshabur et al. (2018)). *Let $f_\Theta : \mathcal{X} \to \mathbb{R}^C$ be any predictor function with parameters Θ and P be a prior distribution on parameters Θ. Then for any $\gamma, m, \delta > 0$, with probability $1-\delta$ over the training set S of size m, for any parameter Θ and any perturbation distribution Q over parameters such that $\mathbb{P}_{U\sim Q}\left[\max_{x\in\mathcal{X}} |f_{\Theta+U}(x) - f_\Theta(x)| \le \frac{\gamma}{4}\right] \ge \frac{1}{2}$, we have*

$$\mathcal{L}_D(f_\Theta) \le \mathcal{L}_{S,\gamma}(f_\Theta) + \sqrt{\frac{2KL(\Theta+U||P) + 1 + \log\frac{m}{\delta}}{2(m-1)}}.$$

The above lemma gives a data-independent deterministic bound which depends on the maximum change of the output function over the domain after a perturbation. We combine the Lemma B.1 with the Theorem 3.2 and prove a bound on the perturbation which leads to the following theorem.

Theorem B.2. *Let norm of the input x an $N\times N$ image whose norm is bounded by B, $f_\Theta : \mathcal{X} \to \mathbb{R}^C$ be the predictor function with parameters Θ which is a DNN of depth d made of convolutional blocks. Then for any margin γ, sample size m, $\delta > 0$, with probability $1-\delta$ over the training set S, any parameter Θ and any $\alpha_i > 0$ such that $\max_{x\in\mathcal{X}} |f_\Theta(x) - f_{\Theta^\alpha}(x)| \le \frac{\gamma}{8}$, we have*

$$\mathcal{L}_D(f_\Theta) \le \mathcal{L}_{S,\gamma}(f_\Theta) + \sqrt{\frac{\sum_{i=1}^d k_i \log\left(1 + \frac{[32edB\alpha_i \left\|\theta_i^F - \theta_i^0\right\|_\mathcal{F} \prod_{i\ne j}\left\|\theta_i^\alpha\right\|_2 \sqrt{\log(4dN^2)}]^2}{c_i\gamma^2}\right) + \log\frac{m}{\delta} + \tilde{O}(1)}{m-1}}, \tag{6}$$

where k_i is the number of parameters in module i. For example, for a convolution module with kernel size $q_i \times q_i$ and number of output channels c_i, $k_i = p^2 c_{i-1} c_i$.

Proof. First, we express the value of the $\tilde{O}(1)$ term in Theorem B.2 , which is equal to ϵ_2 as below

$$\epsilon_2 = 1 + \sum_i \log\left(7m + 2\log\left(\frac{k_i}{k_i\gamma^2/\left(16e\prod_{j\ne i}\|\theta_i^\alpha\|_2 \log(4dN^2)\right)^2 + \alpha_i^2\left\|\theta_i^F - \theta_i^0\right\|_\mathcal{F}^2}\right)\right). \tag{7}$$

We note that for any Θ, Θ', if $\max_{x\in\mathcal{X}} \|f_\Theta(X) - f_\Theta\|_\infty \le \gamma/2$ then $\mathcal{L}(f_\Theta) \le \mathcal{L}_\gamma(f'_\Theta)$. The reason is that the output for each class can change by at most $\gamma/2$ and therefore the label can only change for the data points that are within γ of the margin.

We start using the assumptions on the perturbation bound. Combining the results from Theorems B.1 and 3.2, we can get the following bound.

$$\begin{aligned}
\mathcal{L}_D(f_{\Theta^F}) &\le \mathcal{L}_{D,\frac{\gamma}{4}}(f_{\Theta^\alpha}) \qquad (8)\\
&\le \mathcal{L}_{S,\frac{3\gamma}{4}}(f_{\Theta^\alpha}) + \sqrt{\frac{\frac{1}{2}\sum_{i=1}^d k_i\log\left(1 + \frac{\alpha_i^2\left\|\theta_i^F - \theta_i^0\right\|_\mathcal{F}^2}{k_i\sigma_i^2}\right) + \log\frac{m}{\delta} + \epsilon_2}{m-1}}\\
&\le \mathcal{L}_{S,\gamma}(f_{\Theta^\alpha}) + \sqrt{\frac{\frac{1}{2}\sum_{i=1}^d k_i\log\left(1 + \frac{\alpha_i^2\left\|\theta_i^F - \theta_i^0\right\|_\mathcal{F}^2}{k_i\sigma_i^2}\right) + \log\frac{m}{\delta} + \epsilon_2}{m-1}}
\end{aligned}$$

where ϵ_2 is given above in equation 7.

Therefore, it suffices to find the value of σ_i under which the assumption on norm of perturbation in function space holds and then simplify the following upper bound given the desired value of σ_i.

In order to find the desired value of σ_i we use the following two lemmas. First we adopt the perturbation lemma in (Neyshabur et al., 2018) to bound the change in the output a network based on the magnitude of the perturbation:

Lemma B.3 ((Neyshabur et al., 2018)). *Let norm of input x be bounded by B. For any $B > 0$, let $f_\Theta : \mathcal{X} \to \mathbb{R}^C$ be a neural network with ReLU activations and depth d. Then for any $\Theta, x \in \mathcal{X}$,* *and any perturbation U s.t. $\|u_i\|_2 \leq \|\theta_i\|_2$, the change in the output of the network can be bounded as follows*

$$\|f_{\Theta+U} - f_{\Theta}\|_2 \leq eB \prod_{i=1}^{d} \|\theta_i\|_2 \sum_{j=1}^{d} \frac{\|u_i\|_2}{\|\theta_i\|_2}. \tag{9}$$

We next use the following lemma from (Pitas et al., 2017) that bounds the magnitude of the Gaussian perturbation u_i for each convolutional module based on the standard deviation of the perturbation.

Lemma B.4 ((Pitas et al., 2017)). *Let u_i be a Gaussian perturbation for each module i of a convolutional model. Let N be the image size, q_i , c_i be the kernel size and the number of output channels at module i respectively. We have that*

$$\mathbb{P}\left[\|u_i\|_2 \geq \sigma_i\big(q_i(2\sqrt{c_i}) + t\big)\right] \leq 2N^2 e^{-\frac{t^2}{2q_i^2}}.$$

The above lemma suggests that by taking union bounds over all modules, we can ensure that with probability $1/2$ we have that for any module i, the following upper bound on the spectral norm of the perturbation holds.

$$\begin{aligned}
\|u_i\|_2 &\leq \sigma_i q_i(2\sqrt{c_i} + \sqrt{2\log(4dN^2)}) \\
&\leq 2\sigma_i q_i(\sqrt{c_i} + \sqrt{\log(4dN^2)}) \\
&\leq 4\sigma_i q_i \sqrt{c_i \log(4dN^2)}.
\end{aligned}$$

Combining this with perturbation bound in Equation 9, we have that

$$\begin{aligned}
\|f_{\Theta^\alpha+U} - f_{\Theta^\alpha}\|_2 &\leq eB \sum_{i=1}^{d} \|u_i\|_2 \prod_{j\neq i}^{d} \|\theta_j^\alpha\|_2 \\
&\leq 4eB \sum_{i=1}^{d} \sigma_i q_i \sqrt{c_i \log(4dN^2)} \prod_{j\neq i}^{d} \|\theta_i^\alpha\|_2 \leq \frac{\gamma}{8},
\end{aligned}$$

where the last inequality can be achieved with

$$\sigma_i = \frac{\gamma}{32edB \prod_{i=1}^{d} \|\theta_i^\alpha\|_2 \, q_i \sqrt{c_i \log(4dN^2)}}. \tag{10}$$

Therefore, this value for σ_i ensures the assumption on norm of perturbation in function space in Theorem B.2 holds and hence completes the proof.

Moreover, we show how we get the value of ϵ_2, by showing the simplification from inserting the value for σ_i from Equation equation 10 as follows.

$$\begin{aligned}
\log\left(1 + \frac{\alpha_i^2 \left\|\theta_i^F - \theta_i^0\right\|_{\mathcal{F}}^2}{k_i \sigma_i^2}\right) &\leq \log\left(1 + \frac{\left[\alpha_i \left\|\theta_i^F - \theta_i^0\right\|_{\mathcal{F}}\right]^2}{q_i^2 c_i^2 \sigma_i^2}\right) \\
&\leq \log\left(1 + \frac{[32edB\alpha_i \left\|\theta_i^F - \theta_i^0\right\|_{\mathcal{F}} \prod_{i=1}^{d} \|\theta_i^\alpha\|_2 \, q_i \sqrt{c_i \log(4dN^2)}]^2}{q_i^2 c_i^2 \gamma^2}\right) \\
&= \log\left(1 + \frac{[32edB\alpha_i \left\|\theta_i^F - \theta_i^0\right\|_{\mathcal{F}} \prod_{i=1}^{d} \|\theta_i^\alpha\|_2 \sqrt{\log(4dN^2)}]^2}{c_i \gamma^2}\right).
\end{aligned}$$

Then, ϵ_2 can also be simplified as follows.

$$
\begin{aligned}
\epsilon_2 &= 1 + \sum_i \log\left(7m + 2\log\left(\frac{k_i}{k_i\sigma_i^2 + \alpha_i^2 \left\|\theta_i^F - \theta_i^0\right\|_{\mathcal{F}}^2}\right)\right) \\
&\leq 1 + \sum_i \log\left(7m + 2\log\left(\frac{k_i}{k_i\gamma^2/\left(32edB\prod_{i=1}^d \|\theta_i^\alpha\|_2\, q_i\sqrt{c_i\log(4dN^2)}\right)^2 + \alpha_i^2 \left\|\theta_i^F - \theta_i^0\right\|_{\mathcal{F}}^2}\right)\right) \\
&\leq 1 + \sum_i \log\left(7m + 2\log\left(\frac{k_i}{\gamma^2/\left(32edB\prod_{i=1}^d \|\theta_i^\alpha\|_2 \sqrt{\log(4dN^2)}\right)^2 + \alpha_i^2 \left\|\theta_i^F - \theta_i^0\right\|_{\mathcal{F}}^2}\right)\right) \\
&\leq 1 + \sum_i \log\left(7m + 2\log(k_i) - 4\log\left(\alpha_i \left\|\theta_i^F - \theta_i^0\right\|_{\mathcal{F}} + \gamma/32edB\prod_{i=1}^d \|\theta_i^\alpha\|_2 \sqrt{\log(4dN^2)}\right)\right).
\end{aligned}
$$

□

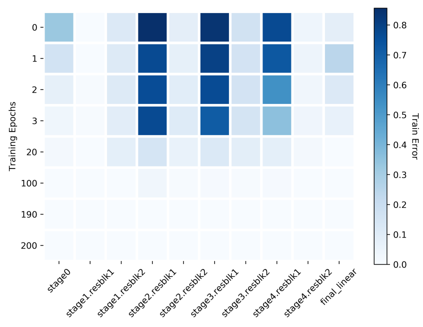
C Details on experimental setup

For all our experiments, we use CIFAR10 dataset. To train our networks we used Stochastic Gradient Descent (SGD) with momentum 0.9 to minimize multi-class cross-entropy loss. Each model is trained until the cross-entropy loss on the training dataset falls below 0.15. The ResNets and VGGs were trained using a stage-wise constant learning rate scheduling with a starting learning rate of 0.1 and with a decrease by a multiplicative factor of 0.2 every 60 epochs. FCN was trained with an initial learning rate of 0.1 with a decrease by a multiplicative factor of 0.2 every 200 epochs. Batch size of 128 was used for all models and weight decay with factor 5e-4 was used to train all networks.

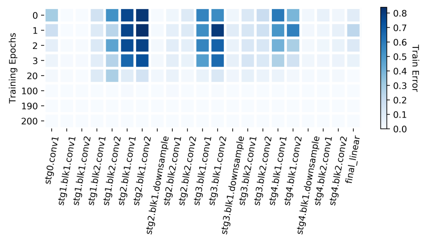
We mainly study three types of neural network architectures:

- Fully Connected Networks (FCNs): The FCNs consist of 2 fully connected layers with 5000 and 1000 hidden units respectively. Each of these hidden layers is followed by a batch normalization layer and a ReLU activation. The final output layer (that follows the ReLU activation in the second layer) has an output dimension of 10 (number of classes).
- VGGs: Architectures from (Simonyan & Zisserman, 2014) that consists of multiple convolutional layers, followed by multiple fully connected layers and a final classifier layer (with output dimension 10). We study the VGG with 16 layers.
- ResNets: Architectures used are ResNets V1 (He et al., 2016). All convolutional layers (except downsample convolutional layers) have kernel size 3×3 with stride 1. Downsample convolutions have stride 2. All the ResNets have five stages (0-4) where each stage has multiple residual/downsample blocks. These stages are followed by a maxpool layer and a final linear layer. Here are further details about the ResNets used in the paper:
  - ResNet18: All ResNet18s studied in the paper (in Table 2 and Figure 5b) have 1 convolutional layer in Stage 0 (64 ouput channels), Stage 1 has 2 residual blocks (64 output channels), Stage 2 has one downsample block and one residual block (128 output channels), Stage 3 has one downsample block and one residual block (256 output channels) and Stage 4 again has one downsample block and a residual block (512 output channels).

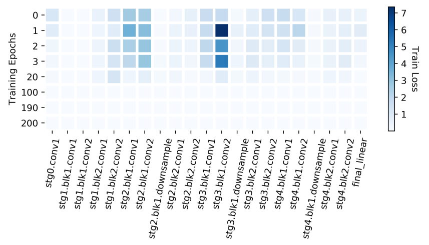
    For the ResNets studied in Table 3, ResNet (1x width) has the same architecture as described above but with 16, 16, 32, 64 and 128 output channels in the stages 0-4.
  - ResNet34: ResNet34 architectures in this paper have 5 stages. Stage 0 has 1 convolutional layer with 64 output channels followed by a ReLU activation. Stage 1 has 3 residual blocks (64 output channels), Stage 2 has 1 downsample block and 3 residual blocks (128 output channels), Stage 3 has 1 downsample block and 5 residual blocks (256 output channels) and, Stage 4 has 1 downsample block and 2 residual blocks (512 output channels).



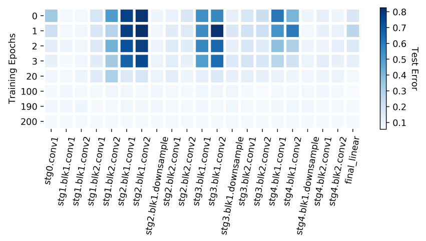
(a) Rewind analysis of (Zhang et al., 2019a)



(b) We rewind each module, whereas (Zhang et al., 2019a) rewind each block



(c) The effect of rewinding on train loss



(d) The effect of rewinding on test error

Figure 6: Analysis of rewinding modules to initialization for ResNet-18 architecture. Each row represents a layer in ResNet18-v1 that has been rewound to the initialization value and each column represents a particular training epoch.

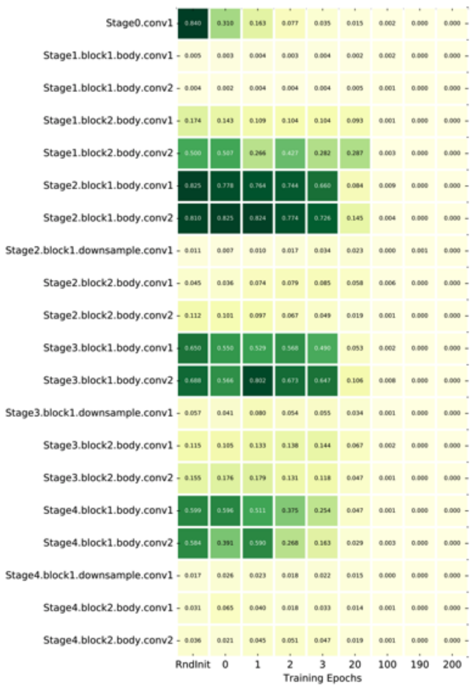
- ResNet101: ResNet34 architectures in this paper again have 5 stages. Stage 0 has 1 convolutional layer with 64 output channels followed by a ReLU activation. Stage 1 has 1 downsample block and 2 residual blocks (256 output channels), Stage 2 has 1 downsample block and 3 residual blocks (512 output channels), Stage 3 has 1 downsample block and 22 residual blocks (1024 output channels) and, Stage 4 has 1 downsample block and 2 residual blocks (2048 output channels).

The ResNets and VGGs in the paper are trained without batch normalization.

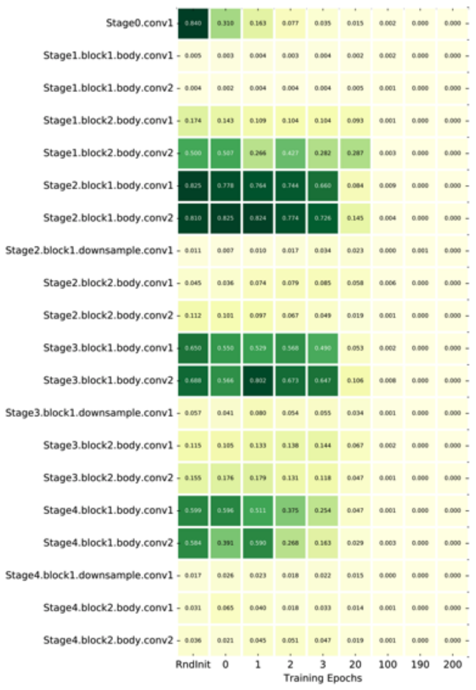
During training on CIFAR10, images are padded with 4 pixels of zeros on all sides, then randomly flipped (horizontally) and cropped. Global mean and standard deviation are computed on all training images and applied to normalize the inputs of CIFAR10.

While training a ResNet18 on the CIFAR10 dataset with 20% of the labels randomly corrupted, we *do not* augment the training set with images that are randomly flipped and cropped. We also do not use weight decay during training these networks.

D Figures



(a) Fixup initialization



(b) Adam optimizer

Figure 7: Criticality pattern of Resnet18-v2 when trained with Fixup initialization, Adam optimizer

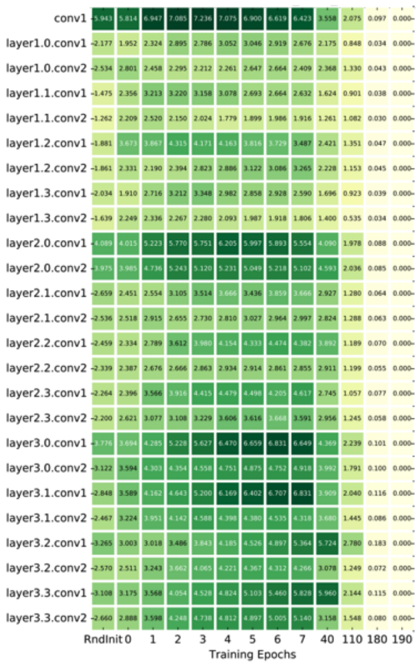
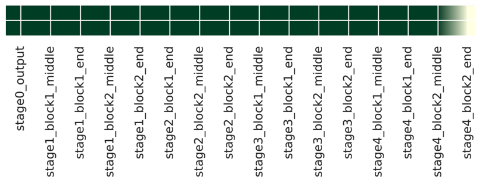
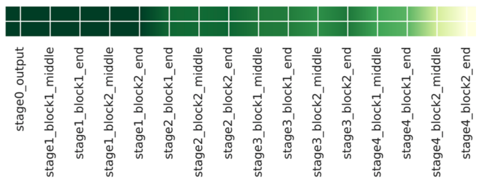


Figure 8: Operator norm of difference from initialization



(a) Rewind an ambient module



(b) Rewind a critical module

Figure 9: Similarity in activation patterns when an ambient or critical module is rewound. Darker green denotes higher similarity

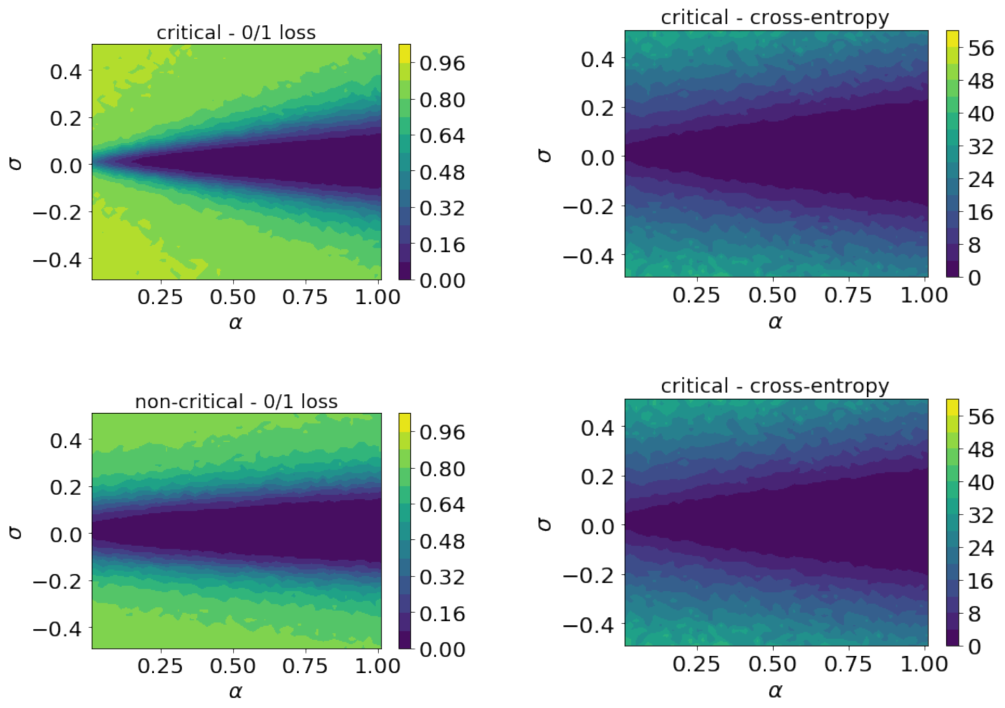


Figure 10: 0/1 loss and cross-entropy loss for critical and non-critical modules, for given different values of σ and α in from Definition 3.1.