- Extensive study of the statistical properties of BN and other normalizers
- Proposes two other normalizers "PreLayerNorm" and "RegNorm" that are as effective as BN but doesn't depend on the batch dimension

Is Batch Norm unique? An empirical investigation and prescription to emulate the best properties of common normalizers without batch dependence

Vinay Rao [1] **Jascha Sohl-Dickstein** [1]

Abstract

We perform an extensive empirical study of the statistical properties of Batch Norm and other common normalizers. This includes an examination of the correlation between representations of minibatches, gradient norms, and Hessian spectra both at initialization and over the course of training. Through this analysis, we identify several statistical properties which appear linked to Batch Norm's superior performance. We propose two simple normalizers – PreLayerNorm, and RegNorm – which better match these desirable properties without involving operations along the batch dimension. We show that PreLayerNorm and RegNorm achieve much of the performance of Batch Norm without requiring batch dependence, that they reliably outperform LayerNorm, and that they can be applied in situations where Batch Norm is ineffective.

1. Introduction

Normalization has a long history in machine learning, and is central to the performance of Deep Neural Networks (DNNs). Traditional whitening transformations (Kessy et al., 2018; Krizhevsky, 2009a) and modern normalizers like Layer Norm (Ba et al., 2016), Batch Norm (Ioffe & Szegedy, 2015) and others (Ulyanov et al., 2016; Wu & He, 2018; Shen et al., 2020; Salimans & Kingma, 2016) have helped train increasingly complex and deep DNN architectures (He et al., 2016; Vaswani et al., 2017). Batch Norm has been particularly effective (Xiao et al., 2018). Despite model performance being highly sensitive to choice of normalizer, there has been relatively little work exploring what properties of normalizers lead to good generalization performance. Although Ioffe & Szegedy (2015) proposed that Batch Norm helps by reducing Internal Covariate Shift (ICS), Santurkar et al. (2018) reported that it does not help ICS, but instead smooths the loss landscape. Bjorck et al. (2018); Luo et al. (2019) both found that Batch Norm allows large learning rates in specific settings, while Kohler et al. (2018) identified situations where Batch Norm accelerates training. Balduzzi et al. (2017) found that Batch Norm in ResNets allows deep gradient signal propagation in contrast to networks without it. Ghorbani et al. (2019) studied the Hessian spectra of networks with Batch Norm and found that it helped prevent the appearance of large isolated eigenvalues in the spectrum. Yang et al. (2019) studied the signal propagation of input representations and gradients through networks with Batch Norm and found that it causes gradient explosion under certain settings, while also causing exponential decay of information in deep networks without residual connections. A frequent theme is an absence of comparison against other normalizers, raising the question of whether these behaviors or advantages are unique to Batch Norm.

Many people don't know about this!

Our contributions can be summarized as follows:

- We study the effect of many normalization techniques on information propagation (Schoenholz et al., 2016; Poole et al., 2016) (Section 4.1), loss surface curvature (Section 6), and early training dynamics (Section 5).
- With extensive empirical evidence, we study the effect of depth, learning rate, and other hyperparameters and the behavior of normalization techniques under these changing settings. We also disentangle the importance of the mean subtraction and standard-deviation operations across the batch-dimension in Batch Norm.
- Directly motivated by observed differences, we propose two simple alternatives to Batch Norm and Layer Norm that can match or outperform their generalization ability while closely matching likely beneficial statistical properties of Batch Norm (Sections 3 and 7).
- We show that these reach similar levels of performance as Batch Norm on datasets like Cifar10 and ImageNet, and improve on Layer Norm (Ba et al., 2016) for Transformer networks (Vaswani et al., 2017) and recurrent architectures without resorting to additional complexity like Power Norm (Shen et al., 2020).
- Finally, we share a set of failed approaches (Section A) we tried which did not improve model performance.

[1]Google Research, Mountain View, CA, USA. Correspondence to: Vinay Rao <vinaysrao@google.com>.

2. Normalization techniques

Let a deep neural network with L layers have inputs x^0 of size (N,H,W,C) or (batch size, height, width, channels). Each layer performs an operation of the form

$$x^l = \phi\left(z^l\right), \qquad z^l = W^l \circ x^{l-1} + b^l, \tag{1}$$

where x^l is the output at layer l, W^l and b^l are layer l's weight and bias parameters, $\circ \in \{\text{Dense, Convolution}\}$ indicates matrix multiplication or convolution, and ϕ is a nonlinear activation function like ReLU or tanh applied elementwise. The normalization operations discussed below modify this layer equation to be

$$x^l = \phi\left(\gamma \tilde{z}^l + \beta\right), \qquad \tilde{z}^l = \frac{z^l - \mu(\cdot)}{\sigma(\cdot)} \tag{2}$$

where the offset $\mu(\cdot)$ and scale $\sigma(\cdot)$ are computed in a different way for each normalizer. γ and β are post-normalization scaling and bias parameters.

Batch Norm (Ioffe & Szegedy, 2015) requires additional overhead and complexity while training to accumulate statistics for test-time, to synchronize these statistics across accelerators while training, and to accumulate statistics over the batch dimension (involving non-local memory-access patterns). Ghorbani et al. (2019) showed that using population mean and variance while training causes the Hessian to be ill-conditioned. We will show that this difference in test and train time behavior is crucial for the performance benefits of Batch Norm (Section 7, Figure 5). For models with sequential inputs and outputs, the correct way to handle different length sequences within a batch, and more generally to handle the temporal axis, is not clear.

One of the reasons Why BN isn't used with RNNs that often

To overcome the difficulty of applying Batch Norm to sequential inputs/outputs, Ba et al. (2016) introduced **Layer Norm**, which removes the dependency on the batch-size and has been used in achieving state of the art results with Transformers (Vaswani et al., 2017; Ho et al., 2019). Weight Norm (Salimans & Kingma, 2016) also removes the interaction between elements in the batch, while aiding good conditioning for optimization.

To disentangle the effects of subtracting the batch mean and the division by the batch standard deviation in our experiments, we consider two variants that ensure consistent scaling: Batch-Mean-Layer-Variance (**BMLV**) and Layer-Mean-Batch-Variance (**LMBV**). Table 1 summarizes these techniques in equation form.

Many other approaches have been proposed as alternatives to Batch Norm, including Group Norm (Wu & He, 2018), Instance Norm (Ulyanov et al., 2016), Power Norm (Shen et al., 2020) that improves performance of Transformer (Vaswani et al., 2017) networks, Batch Renormalization (Ioffe, 2017) that reduces dependency on the minibatch, and Norm Propagation (Arpit et al., 2016; Laurent et al., 2017).

3. Improved normalizers

We propose two new normalizers which better match the statistical properties and good performance of Batch Norm without requiring operations over the batch dimension or differences in test and train time behavior. See Table 1 for additional specification. Their design will be further motivated by the experimental analysis of existing normalizers presented below. Qualitative properties of different normalizers are compared in Table 2.

3.1. Pre-Normalization:PreLayerNorm

A simple shift in the order of normalization operations that we term Pre-Normalization, helps mimic some properties of Batch Norm without batch-axis operations.

Particularly, we study PreLayerNorm, where the normalizer subtracts the layer's mean μ^i before the Affine operation and divides the layer's standard-deviation σ^i before the nonlinearity ϕ.

To see why PreLayerNorm can be expected to behave differently than Layer Norm, consider the affine transformation $z^l = W^l \circ x^{l-1}$ where $W^l \in R^{N_{l-1} \times N_l}$ is initialized from an isotropic Gaussian. Even if x^{l-1} has non-zero mean across units, in expectation z^l will be zero-mean across units, due to the randomness in the weight matrix W^l. Therefore, subtracting the mean across units can be expected to have a far smaller effect if applied after multiplication by W^l. To the degree that the mean across units in x^{l-1} resembles the mean across the batch, Pre-Normalization more closely emulates the behavior of Batch Norm.

3.2. Regularized Normalization (RegNorm)

We propose a normalization scheme that in conjunction with a regularization term added to the loss, ensures that the mean of the activations across the batch is 0 while ensuring consistent scaling. At initialization, this technique mimics the information propagation properties of Layer Norm, while after minimizing the regularizer it mimics those of BMLV. The regularization term is

$$r(\bar{z}) = \mathbb{E}_{a,b}\left[\sum_i ((\bar{z}_i^a + \bar{z}_i^b)^2 - 2)\right], \tag{3}$$

where $a, b \in Batch$ and $i \in (H, W, C)$. Note that the regularizer is computed after the inputs are normalized, and so will not depend on the activation norm. The loss-function used during model training is modified to be $L' = L + \lambda \sum_l r^l$, where the sum is over network layers, and λ is a regularization strength hyperparameter. This regularizer has a unique minimum when the mean of the activations across

Not sure how well this will play with other regularization like Dropout, Spatial Dropout, L2, etc

Normalization	Layer Op	Norm Op	Mean Axes	Std.Dev Axes
Batch Norm	$x^l = \phi\left(\gamma\tilde{z}^l + \beta\right)$	$\tilde{z}^l = \frac{z^l - \mu^B}{\sigma^B}$	$\mu^B : (N, H, W)$	$\sigma^B : (N, H, W)$
Layer Norm	$x^l = \phi\left(\gamma\tilde{z}^l + \beta\right)$	$\tilde{z}^l = \frac{z^l - \mu^i}{\sigma^i}$	$\mu^i : (H, W, C)$	$\sigma^i : (H, W, C)$
Weight Norm	$x^l = \frac{W^l \circ x^{l-1}}{\|\|W^l\|\|_F} + b^l$	-	-	-
(Ours) BMLV	$x^l = \phi\left(\gamma\tilde{z}^l + \beta\right)$	$\tilde{z}^l = \frac{z^l - \mu^B}{\sigma^i}$	$\mu^B : (N, H, W)$	$\sigma^i : (H, W, C)$
LMBV	$x^l = \phi\left(\gamma\tilde{z}^l + \beta\right)$	$\tilde{z}^l = \frac{z^l - \mu^i}{\sigma^B}$	$\mu^i : (H, W, C)$	$\sigma^B : (N, H, W)$
(Ours, improved) PreLayer Norm	$x^l = \phi\left(\gamma\hat{z}^l + \beta\right)$	$\hat{z}^l = \frac{\bar{z}^l}{\sigma^i(\cdot)}$, $\bar{z}^l = W^l \circ \left(x^{l-1} - \mu^i(x^{l-1})\right)$	$\mu^i : (H, W, C)$	$\sigma^i : (H, W, C)$
RegNorm*	$x^l = \phi\left(\gamma\bar{z}^l + \beta\right)$	$\bar{z}^l = \frac{z^l}{\sigma^i(\cdot)}, z^l = W^l \circ x^{l-1}$	-	$\sigma^i : (H, W, C)$

Table 1: Specifications for the the normalization schemes we consider in this paper. *RegNorm additionally uses a regularization term $r(\bar{z}) = \mathbb{E}_{a,b}\left[\sum_i((\bar{z}_i^a + \bar{z}_i^b)^2 - 2)\right]$ where $a, b \in Batch$ and $i \in (H, W, C)$ (Section 3.2).

the batch is 0 – see Appendix B for a proof. We further support this empirically in Figure App.11.

4. Properties at initialization

In this section we analyze the statistical properties of networks at initialization, under different normalization techniques. Xiao et al. (2019) studied the effect of information propagation on trainability and generalization, showing that while networks in the chaotic phase (initially similar samples become dissimilar with depth) train faster, their generalization suffers. Conversely, networks in the ordered phase (the correlation between samples converges to 1 with depth) are harder to train, but show better generalization. Normalization techniques induce different information propagation behavior. We first examine this behavior at initialization. In Section 5, we will see how these phases evolve quickly through the early stages of training.

4.1. Information propagation through layers

Information propagation (Schoenholz et al., 2016; Poole et al., 2016) is the measure of how similar the representations of two inputs are through a DNN. We calculate this as the correlation between two similar minibatches – $\text{corr}(x + \epsilon_1, x + \epsilon_2)$, where $\epsilon_1, \epsilon_2 \sim \mathcal{N}(0, \sigma)$, and x here represents an entire minibatch of data. If the correlation approaches 1 after the application of many layers, then dissimilar minibatches are converging on the same activation patterns. On the other hand if the correlation converges towards 0 (or other value less than 1), then the network is chaotic or behaves like a hash map, where inputs with initially minor differences are mapped to very different points in the representation space.

We first plot the propagation of MNIST inputs through a fully-connected network with ReLU activations in Figure 1(a). In Figure 2(a) we consider a WideResnet (Zagoruyko & Komodakis, 2016) with 26 layers and no skip connections, with a channel multiplier set to 2, and propagate batches from the Cifar-10 dataset (Krizhevsky, 2009b). The majority of experiments in the paper will use this WideResnet architecture, with experiments including skip connections in the Appendix.

We find that Batch Norm preserves less information about the input than any other standard normalization technique. Skip connections improve information propagation (Figure App.1), but don't change the relative behavior of Batch Norm compared with other normalizers. The BMLV and LMBV conditions suggest that for a ReLU network the subtraction of the mean across the batch is the primary driver of correlation decay, and division by the standard deviation across the batch plays little role.

4.2. Correlation of gradients through layers

We examine the correlation of gradients through the same fully-connected network in Figure 1(b). Sankararaman et al. (2019) postulate that gradient confusion, or highly-correlated gradients causes slower training. We notice that Batch Norm produces the most decorrelated gradients, while Layer Norm produces highly-correlated gradients.

- To measure information propagation, the correlation between two mini-batches is calculated as they propagate through the layers of a DNN.
- A correlation value around zero would suggest that even if the inputs differ by only a very very small amount, they get mapped into a completely representation.
- BN preserves less information about the input than any other normalizer. Skip connections help in information propagation but doesn't change the behaviour of BN. Findings suggest that the mean subtraction in a ReLU network is the main cause of correlation decay

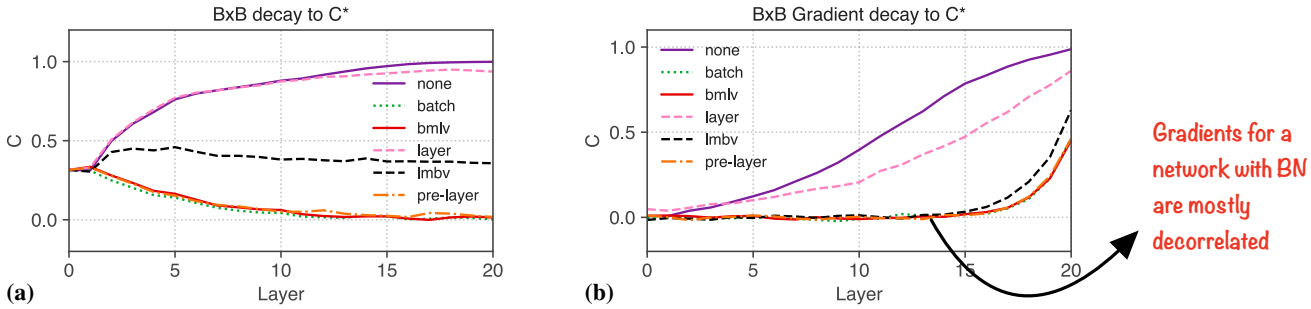

Figure 1: Correlation of input representations *(a)* and gradients *(b)* between two similar minibatches through 20 layers of a fully-connected DNN with ReLU activation. *(a)* Batch Norm decorrelates the representations through depth (as do BMLV, PreLayerNorm) while LMBV preserves the correlation. The representations become highly correlated for Layer Norm and No Norm. *(b)* While all normalizers cause gradients to become decorrelated at the first layer, Batch Norm, BMLV and PreLayerNorm cause faster decorrelation.

There is a spike for all norms except for BN and BMLV. And the spike is more or less at a certain depth. Why?

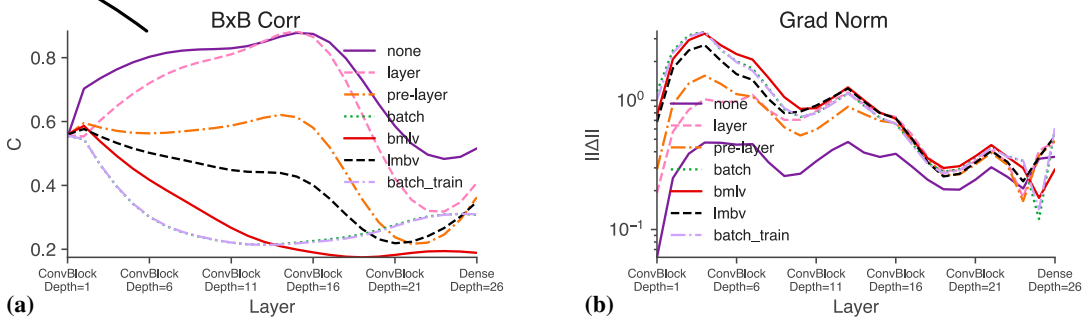

Figure 2: Correlation of inputs *(a)* and L2-Norm of gradients *(b)* between similar minibatches through a 26-layer WideResnet with ReLU activations and no skip connections. *(a)* Batch Norm (and BMLV) decorrelates the representations through depth while Layer Norm leads to highly correlated representations. Using PreLayerNorm leads to more decorrelation and greater similarity to Batch Norm. *(b)* Batch Norm (and BMLV) induces the most gradient explosion at initialization, while No Norm induces the least. PreLayerNorm induces more gradient explosion than Layer Norm.

4.3. Gradient explosion through layers

Yang et al. (2019) predicted that Batch Norm causes gradient explosion in DNNs, when repeatedly applied in series. We empirically observe in Figure App.2 that with skip connections, this phenomenon does not significantly occur (note that this *is* consistent with the theory result). However, without skip connections (Figure 2(b)), we observe that Batch Norm and closely related techniques do cause more growth in gradients with depth compared to other techniques.

5. Early training dynamics

Lewkowycz et al. (2020); Frankle et al. (2020); Jastrzebski et al. (2020) observed that DNNs undergo important changes in the early stages of training. We examine the behavior of the properties from Section 4 over the course of training, to understand how the effect of different normalizers evolves over the course of training.

Using Batch Norm as a yardstick, we see how our proposed techniques compare to its early training dynamics. Unless explicitly stated, we plot the dynamics for DNNs without skip/residual connections (see Appendix C for experiments with residual connections). We perform a grid search $[10^{-3}, 10^{-2}, \ldots, 10]$ over learning rates for each of the approaches while generating these plots. We also discuss the surprising transition of Batch Normalization away from inducing chaos at initialization – underscoring the importance of the first few steps of training.

From Figure 3(a), we observe that **accuracy trajectories** look similar for most normalizers, with layer norm lagging other approaches. By step 1000, networks with all normalization schemes perform with greater than 50% accuracy.

We plot the **correlations of the output**s at the final pre-activation layer between similar batches as described in Section 4.1 through training (Figure 3(b)). Even though Batch Norm is chaotic at initialization where all inputs are mapped to very different points in representation space, there is a phase where the representations become more correlated, before gradually decorrelating again. Additionally, Layer Norm causes its input representations to be highly correlated through training. This might prevent it from exploring rep-

- BN and related techniques cause more growth in the gradients with the depth but why?
- Because BN decorrelate the gradients, hence the training would be faster. But With BN representations are more chaotic in the beginning and gradually becomes correlated (for similar batches) as the training progress and then decorrelates again? More imp question is "when" during the training does this happen? And why does the network decorrelate the representations if they are very similar inputs?

- One thing that isn't' clear here is that whether the batches contain augmented images or not
- Every technique, at some point in the early training, causes the gradients to explode. Is there a way to avoid it? Or do we really want to avoid it?

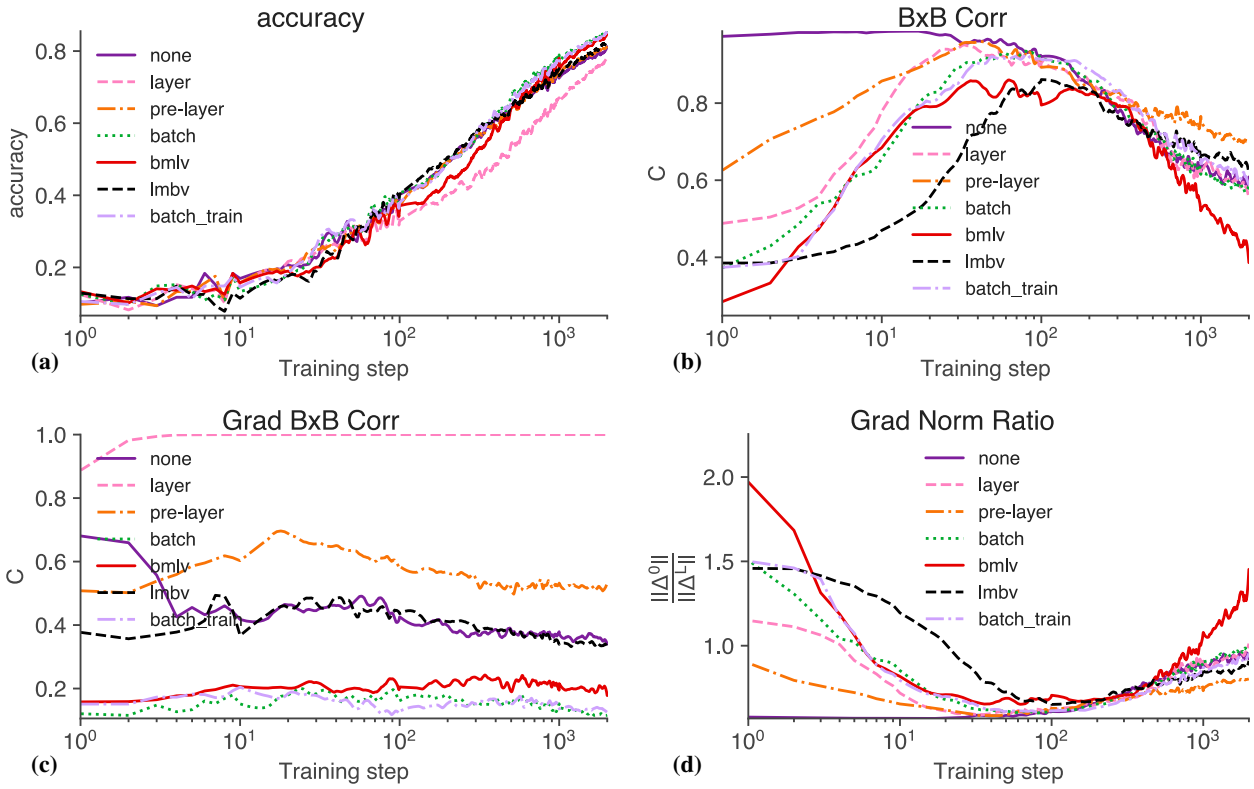

Figure 3: *(a)* Accuracy, *(b)* correlation of output representations, *(c)* correlation of input gradients, and *(d)* ratio of L2-Norm of the gradients at the first layer to the last layer of a 26-layer WideResnet with ReLU activations and no skip connections through the first 2000 steps of training on Cifar10, with each technique trained using tuned learning rates. *(b)* In networks with any normalization, DNNs undergo a transition from being more chaotic (with Batch Norm and BMLV the most chaotic) with decorrelated representations, to more ordered with correlated representations. The correlation between representations then gradually decreases out to 2000 training steps. Note that No Norm only undergoes one transition from highly correlated to decorrelated. *(c)* Batch Norm (and BMLV) has the least correlated gradients or gradient confusion, while Layer Norm has the most highly correlated gradients. PreLayerNorm in this case does not fully match Batch Norm's behavior. *(d)* Batch Norm and BMLV induce the most gradient explosion in the first phase of early training while Layer and PreLayerNorm cause lower gradient explosion. No Norm causes the gradients to vanish. All the techniques undergo a shift where there is an increasing amount of gradient explosion later in the first 2000 training steps.

resentation spaces. Using our proposed Pre-Normalization technique, we are able to get closer to Batch Norm's behavior without any operations across the batch-dimension.

Sankararaman et al. (2019) proposed the concept of **Gradient Confusion**, where having highly similar gradients between batches leads to slower training with SGD. We note from Figure 3(c) that Batch Norm induces the least gradient confusion. However, BMLV behaves similarly. Further, while Layer Norm has **gradient correlations** that rapidly approach 1, PreLayerNorm maintains intermediate values of gradient correlation.

6. Hessian spectrum analysis

In this section, we highlight some properties of the Hessian spectrum of the WideResnet without skip connections described before. We defer the DNN with skip connections to the appendix (Figure App.5) because it shows qualitatively similar relationships between normalizers. We estimate the Hessian for the cross-entropy loss using the Lanczos algorithm as described in Ghorbani et al. (2019) using 512 samples of Cifar-10.

We visualize the **density of eigenvalues** of the Hessian 50 steps into training in Figure 4. We observed that very quickly through training, the spectrum undergoes a transformation (and we note that this is predictive of its shape very late in training), where Batch Norm suppresses its outliers

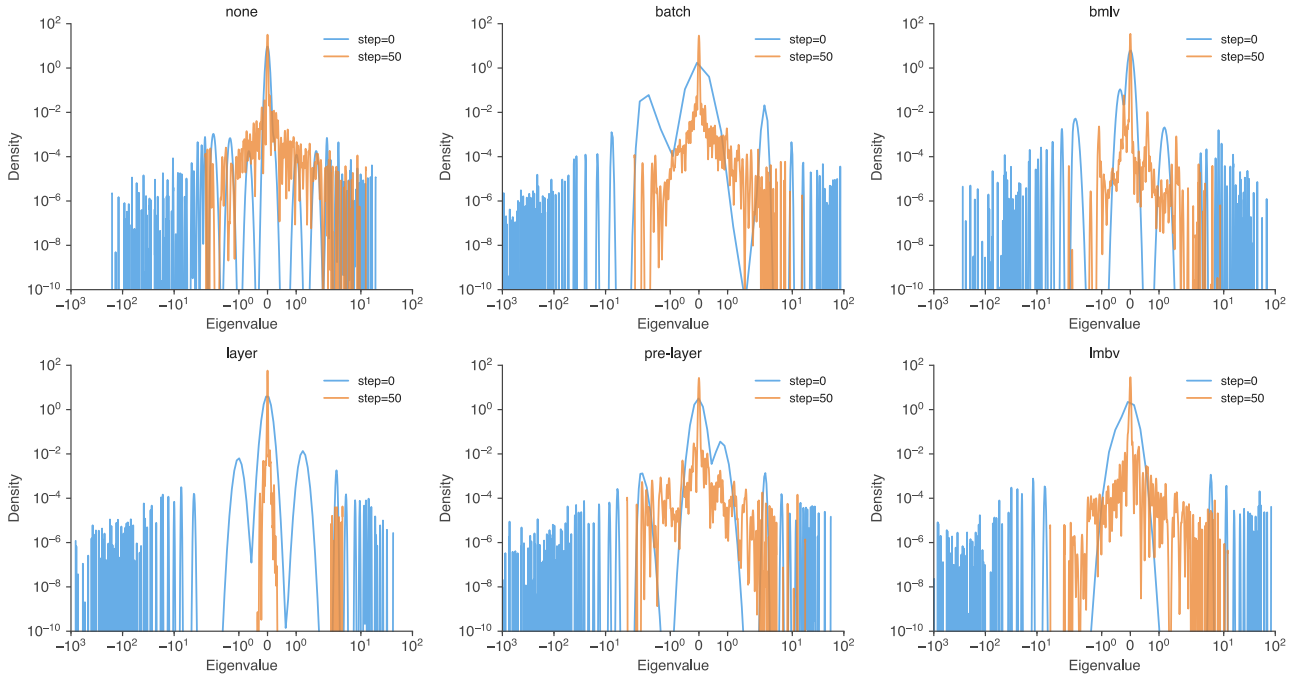

Figure 4: Density of Eigenvalues of the Hessian (cross-entropy) at initialization and 50 steps into training (we used the same learning rate for all techniques) for a 26-layer WideResnet with ReLU activations and no skip connections. Batch Norm has a wider spectrum at initialization. After 50 steps, Layer Norm's spectrum has shrunk and contains outliers. PreLayerNorm behaves similarly to Batch Norm.

and does not have large negative eigenvalues. Layer Norm has a small spectrum, with many outliers which has been noted (Ghorbani et al., 2019) to cause slow training. Our proposal PreLayerNorm is able to condition the Hessian similarly to Batch Norm.

Santurkar et al. (2018) proposed that Batch Norm helps optimization by making the loss-surface smoother. Further, according to Ghorbani et al. (2019) Batch Norm helps by suppressing outliers in the spectrum. This is measured by the **ratio** of the largest eigenvalue (λ_1) to the K-th eigenvalue (λ_k), $\boldsymbol{\lambda_1/\lambda_k}$. We set K=10 to correspond to the number of output classes in Cifar-10, but it should be noted that this is a heuristic guess. We notice from Figure 6(a) that in fact most normalization techniques are able to suppress outliers, except Layer Norm. This might explain the discrepancy in training convolutional architectures with Layer Norm. However, we are able to overcome this with PreLayerNorm without any operations across the batch-axis.

The largest stable learning rate is roughly inversely proportional to the **largest eigenvalue** of the Hessian (LeCun et al., 1992) (see Lewkowycz et al. (2020) for subtleties). At initialization, it seems from Figure 6(b) that Batch Norm must allow the lowest learning rate, however in conjunction with suppressing outliers as seen previously, the maximum eigenvalue quickly decreases, allowing the possibility of larger learning rates. We further note that most other regularizers condition the spectrum similarly through training. LayerNorm is unable to suppress outliers and has a small spectrum that does not allow one to take advantage of its low eigenvalues to use larger learning rates.

7. Performance on realistic tasks

We compare RegNorm and PreLayerNorm against Batch Norm and Layer Norm on several existing tasks. Additional experiments are discussed in Appendix C.

In all experiments, we replace the default normalization technique used with our alternatives, and perform identical hyperparameter tuning.

Table 3 shows the performance of a 26-layer WideResnet (Zagoruyko & Komodakis, 2016) with channel-multiplier set to 10 on Cifar-10. The best performing technique, Batch Norm is highlighted in **bold** while the best amongst our proposed techniques is underlined. Note that our techniques are able to perform better than Batch Norm if it does not make use of aggregated mean and variance during test-time (Batch-Train). Further, using Pre-Normalization helps increase performance.

We further compare the performance of these techniques

How outliers in eight values are checked?

Why LayerNorm doesn't perform as good as BN with CNNs?

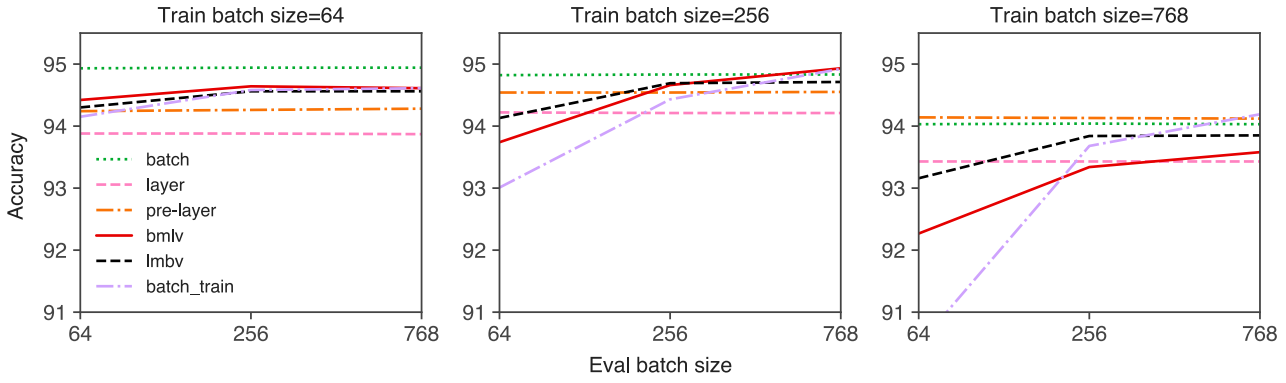

Figure 5: Performance of networks trained with different batch sizes, across a variety of test-time batch sizes. For LMBV, BMLV and Batch-Train, we did not use accumulated statistics. We kept the learning rate constant across batch-sizes for this experiment. Without accumulated statistics (i.e Batch-Train), Batch Norm's performance is highly dependent on batch size. PreLayerNorm outperforms Layer Norm in all the cases.

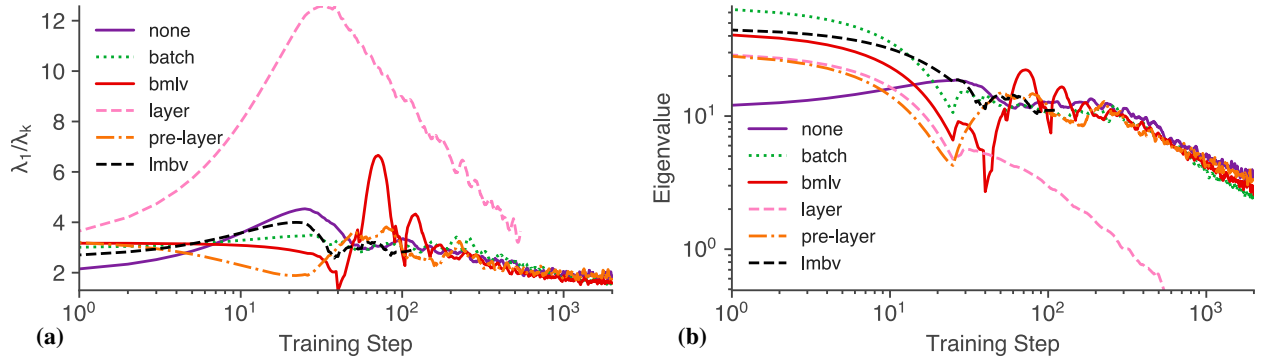

Figure 6: Ratio of the largest eigenvalue of the Hessian (cross-entropy) to the K-th Eigenvalue *(a)* and the largest eigenvalue *(b)* through training a 26-layer WideResnet with ReLU activations and no skip connections. We set K=10 to correspond to the 10 output classes used in Cifar10. *(a)* All techniques including no normalization are able to suppress outlying eigenvalues, except for Layer Norm. *(b)* Although the different techniques differ at initialization, they seem to be scaled and conditioned similarly later in training.

using the Resnet-V1 (He et al., 2016) architecture on ImageNet. Table 3 shows the results with 50 and 101 layers. Although Batch Norm performs best, we close the gap to it without any additional computation and consistently outperform Batch-Train. Further we note that using Pre-Normalization helps increase performance.

Using a Transformer architecture proposed in Ho et al. (2019), we compare PreLayerNorm against Layer Norm and no normalization. Note that in the case of sequential inputs or outputs, operations across the batch-axis are confounding because they introduce the need to use masking and limit the ways in which mean and variance can be aggregated for all the timesteps. There has been previous discussion of the placement of Layer Norm in these architectures in Xiong et al. (2020). By simply replacing Layer Norm with Pre-Normalization, we are able to outperform it on the LM1B language modeling task (Table 3).

From Figures 1, 2 we see that BMLV most closely matches Batch Norm's statistical properties at initialization, and through Figures 3(a,b,c,d) we see again that BMLV most closely resembles Batch Norm's statistical and gradient dynamics through the early stages of training. For these networks with ReLU nonlinearities, we notice that the Batch Mean subtraction is more important, further motivating the design of RegNorm which emulates the behavior of BMLV in a trained network.

8. Batch Norm and dependence on batch-size

With increasing batch size, Batch Norm causes slower decorrelation of input representations through depth at initialization (Yang et al., 2019). The stochastic choice of batch members induces noise in the Batch Norm estimation of the mean and variance. This causes discrepancy in the training and test-time behavior. As we observe in Figure 5, the

Why increasing batch size simply don't give better results?

Desired Property	Batch Norm	BMLV	Layer	PreLayerNorm	RegNorm
Consistent scaling and centering	+	+	+	+	+
Faster training and better generalization	+	+	-*	+	+
Allow training deeper networks	+	+	+	+	+
Outlier eigenvalue suppression through training	+	+	-	+	+
No dependence on batch size	-	-	+	+	+
Consistent training and test behavior	-	-	+	+	+
Easy application to sequential or recurrent models	-	-	+	+	+
Reduce gradient confusion through training	+	+	-	∼	∼
Transition from chaotic to ordered during early training	+	+	+	+	+
Induces gradient explosion†	+	+	-	-	-

Table 2: Several desirable properties for normalizers on DNNs are listed in this table, alongside the methods compared in this paper. A '+' indicates that the technique induces the listed effect, while '-' indicates that it does not.
* Layer Norm enables better generalization in Transformer networks, but does not have the same effects in Deep Convolutional Networks in our experiments. † It remains unclear whether or not this is a desirable property. ∼ While PreLayerNorm and Regularized Norm are able to reduce gradient confusion, they do not do so to the extent that Batch Norm does.

Still unanswered 😕😕

Norm Type	**Cifar10** Wide-Resnet (Accuracy)	**ImageNet** Resnet50-v1 (Accuracy)	**ImageNet** Resnet101-v1 (Accuracy)	**LM1B** Transformer-1D (Perplexity)
None	0.0	71.254	73.153	59.22
Batch	**95.953**	**76.293**	**78.104**	-
Batch-Train	95.512	74.780	76.693	-
Layer	95.653	66.623	75.745	38.341
PreLayerNorm	95.733	75.107	77.655	**38.002**
RegNorm	95.843	74.782	76.725	-
PreRegNorm*	95.813	75.878	77.452	-
BMLV	95.432	75.174	76.814	-
LMBV	95.392	74.837	76.087	-

Table 3: Performance using different normalizers of a WideResnet(26,10) on Cifar10, Resnet50-v1 on ImageNet, Resnet101-v1 on ImageNet and a Transformer1D on LM1B.
* PreRegNorm is defined as RegNorm with layer mean subtraction before the Affine operation.

performance of Batch Norm and other methods that have operations across the batch change with the batch size if the population mean and variance are not aggregated for use at test-time. Even for large test-time batch sizes, Batch Norm without aggregated means and variances produces significantly worse test accuracy (compare $\text{batch}_\text{t}\text{rain}$ to batch). The presence of batch noise during training, and accumulated statistics at test time, seems to be crucial to Batch Norm's better performance than other methods.

9. Discussion

We performed an empirical study of Batch Norm and competing normalizers, examining information propagation and Hessian spectra both at initialization and throughout training. From analyzing the loss surface curvature, we observe that all normalization schemes (except Layer Norm) are able to suppress outlying eigenvalues similarly. While Weight Norm does not affect the conditioning of the Hessian, introducing residual/skip connections improves conditioning similarly across all techniques. Batch Norm induces poor conditioning while training if true aggregated mean and variance are used while training (Ghorbani et al., 2019). However, as we empirically observed previously (Sec 7), it also relies on these aggregated metrics to outperform other techniques while testing. This hints that it is relying on the regularization induced by the noise in the mean and variance estimates. Note in Section A that our attempts to artificially inject regularizing noise did not improve performance. We disentangled the effect of the mean subtraction and standard-deviation division operations by studying BMLV and LMBV and showed that batch mean subtraction plays a bigger role than batch standard-deviation division.

We further introduce two new normalizers – RegNorm and PreLayerNorm – that better match the statistical and conditioning properties of Batch Norm without relying on oper-

ations across the batch. These normalizers perform nearly as well as Batch Norm in cases where Batch Norm is commonly used. We compare these techniques against Batch Norm in Table 2. These new techniques further outperform Layer Norm for models such as Transformers, with sequential inputs/outputs, where Batch Norm is ineffective. Additionally, PreLayer Norm shows better Hessian conditioning in cases where using Layer Norm does not.

As a service to future researchers, we provide a summary of failed experiments in Appendix A.

Acknowledgments

We thank Greg Yang, Justin Gilmer, David Page, Jeffrey Pennington, and Sam Schoenholz for useful discussion and feedback.

References

Arpit, D., Zhou, Y., Kota, B. U., and Govindaraju, V. Normalization propagation: A parametric technique for removing internal covariate shift in deep networks. *ArXiv*, abs/1603.01431, 2016.

Ba, J., Kiros, J. R., and Hinton, G. E. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

Balduzzi, D., Frean, M., Leary, L., Lewis, J. P., Ma, K. W.-D., and McWilliams, B. The shattered gradients problem: If resnets are the answer, then what is the question? *ArXiv*, abs/1702.08591, 2017.

Bjorck, J., Gomes, C. P., and Selman, B. Understanding batch normalization. In *NeurIPS*, 2018.

Frankle, J., Schwab, D. J., and Morcos, A. S. The early phase of neural network training. *ArXiv*, abs/2002.10365, 2020.

Ghorbani, B., Krishnan, S., and Xiao, Y. An investigation into neural net optimization via hessian eigenvalue density. In *ICML*, 2019.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Ho, J., Kalchbrenner, N., Weissenborn, D., and Salimans, T. Axial attention in multidimensional transformers. *ArXiv*, abs/1912.12180, 2019.

Ioffe, S. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. *ArXiv*, abs/1702.03275, 2017.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ArXiv*, abs/1502.03167, 2015.

Jastrzebski, S., Szymczak, M., Fort, S., Arpit, D., Tabor, J., Cho, K., and Geras, K. J. The break-even point on optimization trajectories of deep neural networks. *ArXiv*, abs/2002.09572, 2020.

Kessy, A. M. N., Lewin, A., and Strimmer, K. Optimal whitening and decorrelation. *The American Statistician*, 72:309 – 314, 2018.

Kohler, J. M., Daneshmand, H., Lucchi, A., Zhou, M., Neymeyr, K., and Hofmann, T. Towards a theoretical understanding of batch normalization. *ArXiv*, abs/1805.10694, 2018.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009a.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009b.

Laurent, C., Ballas, N., and Vincent, P. Recurrent normalization propagation. In *ICLR*, 2017.

LeCun, Y., Simard, P. Y., and Pearlmutter, B. A. Automatic learning rate maximization by on-line estimation of the hessian's eigenvectors. In *NIPS 1992*, 1992.

Lewkowycz, A., Bahri, Y., Dyer, E., Sohl-Dickstein, J., and Gur-Ari, G. The large learning rate phase of deep learning: the catapult mechanism, 2020.

Luo, P., Wang, X., Shao, W., and Peng, Z. Towards understanding regularization in batch normalization. *ArXiv*, abs/1809.00846, 2019.

Poole, B., Lahiri, S., Raghu, M., Sohl-Dickstein, J., and Ganguli, S. Exponential expressivity in deep neural networks through transient chaos. In *NIPS*, 2016.

Salimans, T. and Kingma, D. P. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *ArXiv*, abs/1602.07868, 2016.

Sankararaman, K. A., De, S., Xu, Z., Huang, W. R., and Goldstein, T. The impact of neural network overparameterization on gradient confusion and stochastic gradient descent. *ArXiv*, abs/1904.06963, 2019.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. How does batch normalization help optimization? In *NeurIPS*, 2018.

Schoenholz, S. S., Gilmer, J., Ganguli, S., and Sohl-Dickstein, J. Deep information propagation. *arXiv preprint arXiv:1611.01232*, 2016.

Shen, S., Yao, Z., Gholami, A., Mahoney, M., and Keutzer, K. Rethinking batch normalization in transformers, 2020.

Singh, S. and Krishnan, S. Filter response normalization layer: Eliminating batch dependence in the training of deep neural networks. *ArXiv*, abs/1911.09737, 2019.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. S. Instance normalization: The missing ingredient for fast stylization. *ArXiv*, abs/1607.08022, 2016.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.

Wu, Y. and He, K. Group normalization. In *ECCV*, 2018.

Xiao, L., Bahri, Y., Sohl-Dickstein, J., Schoenholz, S. S., and Pennington, J. Dynamical isometry and a mean field theory of cnns: How to train 10, 000-layer vanilla convolutional neural networks. *ArXiv*, abs/1806.05393, 2018.

Xiao, L., Pennington, J., and Schoenholz, S. S. Disentangling trainability and generalization in deep learning. *ArXiv*, abs/1912.13053, 2019.

Xiong, R., Yang, Y., He, D., Zheng, K., xin Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L.-W., and Liu, T.-Y. On layer normalization in the transformer architecture. *ArXiv*, abs/2002.04745, 2020.

Yang, G., Pennington, J., Rao, V., Sohl-Dickstein, J., and Schoenholz, S. S. A mean field theory of batch normalization. *ArXiv*, abs/1902.08129, 2019.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *ArXiv*, abs/1605.07146, 2016.

A. Failed experiments

In addition to the analysis and successful techniques we introduced above, we tried several other seemingly sensible approaches to match the statistical properties of Batch Norm. The following approaches failed to improve performance.

Artificial gradient explosion We have seen that Batch Norm causes more gradient explosion with ReLU networks than other techniques and this was also studied by Yang et al. (2019). To simulate this, we forcefully insert a scaling factor at each layer to increase the scale of the gradients through the network (Figure App.9). Through training, there is a phase where gradient explosion quickly reduces in the early stages. We mimicked this by linearly decreasing the scaling factor until 50 steps and then turning it off (Figure App.10). However, both these techniques failed to improve performance. In fact, we note that the best performing scaling was the default.

Simulating batch noise Batch Normalization seems to rely on the regularization effects introduced by the differences in training and test-time behavior for better generalization. These differences are mainly due to the noise in estimating the mean and variance for every batch while training. We tried to mimic this by adding noise to the inputs after normalization – i.e setting pre-activations to $\bar{z}^l + \epsilon$ where $\epsilon \in \mathcal{N}(0, \sigma)$. Further, we used the same random seed through the network while sampling ϵ, to capture the dependencies between minibatch noise at different layers. This failed to improve generalization during test-time.

Artificial chaos From the observed informational propagation properties of Batch Norm, one could imagine designing architectures that induce chaos at initialization the same way Batch Norm does, and we tried these methods:

- Signed ReLU activation:
  $\phi(x) = \alpha * \frac{1 + sgn(x)}{2} + max(0, x)$
- Normalized Affine Operation:
  $x^l = \phi\left(\frac{Wx}{\sqrt{W^2 x^2}}\right)$
- TLU (Thresholded Linear Unit) proposed by Singh & Krishnan (2019)

Targeting Hessian conditioning To condition the Hessian very similarly to Batch Norm without using it every layer, we used batch mean subtraction and/or batch variance division operations only at the final logits layer[1]. Although this conditions the Hessians similarly, we noticed a drop in generalization.

Regularized learning of variance One of our proposed normalization schemes, RegNorm, also lends itself to a natural extension where with another regularizer ($r(y) = \mathbb{E}_{a,b}\left[\sum_i (z_i^a z_i^b - 1)\right]$), one might push the variance of the batch to be 1 and totally avoid the cost of normalization during inference. This caused training to be unstable and did not improve generalization.

B. Regularizer is minimized at batch-mean 0

The regularizer we propose (Section 3.2) is of the form:

$$r(\bar{z}) = \mathbb{E}_{a,b}\left[(\sum_i (\bar{z}_i^a + \bar{z}_i^b)^2 - 2)\right]$$

where $\bar{z} = \frac{z^l}{\sigma^i(\cdot)}$, $a, b \in Batch, i \in (H, W, C)$. Let N_l be the number of neurons at layer l (W*H*C).

$$\begin{aligned} r(\bar{z}) &= \mathbb{E}_{a,b}\left[\sum_i ((\bar{z}_i^a)^2 + (\bar{z}_i^b)^2 + 2\bar{z}_i^a \bar{z}_i^b - 2)\right] \\ &= 2\mathbb{E}_a\left[\sum_i ((\bar{z}_i^a)^2)\right] + 2\sum_i \mathbb{E}_{a,b}[\bar{z}_i^a \bar{z}_i^b] - 2N_l \end{aligned}$$

Note that $\bar{z_i^a} = \frac{z_i^a}{\sqrt{(1/N_l)\sum_j (z_j^a)^2}}$, and $\mathbb{E}_{a,b}[\bar{z}_i^a \bar{z}_i^b] = \mathbb{E}_a[(\bar{z}_i^a)^2]$

$$\begin{aligned} r(\bar{z}) &= 2\mathbb{E}_i\left[\sum_i \left(\frac{(z_i^a)^2}{(1/N_l)\sum_j (z_j^a)^2)}\right)\right] + 2\sum_i \mathbb{E}[(\bar{z}_i^a)^2] \\ &\quad - 2N_l \\ &= 2N_l + 2\sum_i \mathbb{E}_a[(\bar{z}_i^a)^2] - 2N_l \\ &= 2\sum_i \mathbb{E}_a[(\bar{z}_i^a)^2] \end{aligned}$$

Setting $r(\bar{z})$ to 0, we see that this is only achieved when $\mathbb{E}_a[(\bar{z}_i^a)^2]$ is 0 $\forall a \in Batch$ i.e, when the means of all the inputs the batch are 0, as required.

C. Further properties and ablation studies

[1] Hinted at by https://myrtle.ai/learn/how-to-train-your-resnet-7-batch-norm/

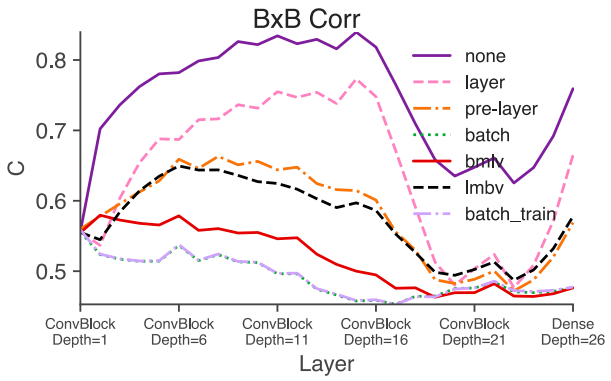

Figure App.1: Correlation of inputs between batches through a 26-layer WideResnet with skip connections. Batch Norm cases the input representations to be more decorrelated through depth than other techniques. However, with skip-connections, inputs are never completely decorrelated.

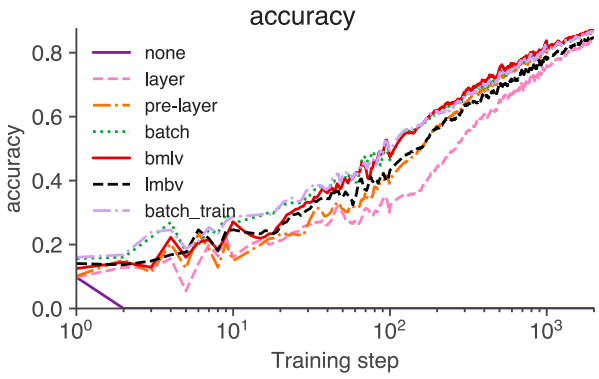

Figure App.3: Accuracy of a 26-layer WideResnet ReLU network with skip connections through the first 2000 steps of training on Cifar10. Note that No Norm fails to train at this depth with high learning rates, and most normalization schemes are able to reach 50% accuracy by 500 steps into training.

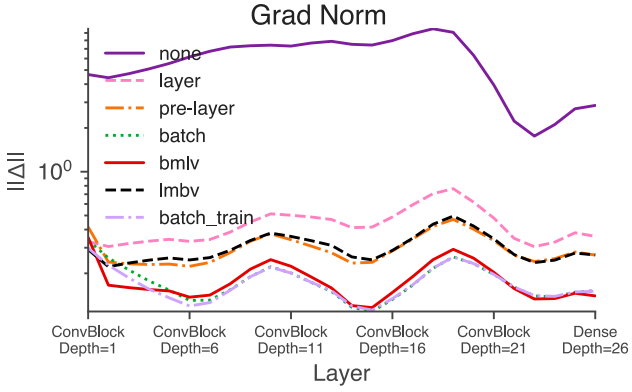

Figure App.2: L2-Norm of the gradients of the parameters of all the layers in a WideResnet without skip connections. With skip connections, all the normalization techniques avoid gradient explosion with No Norm causing the most explosion.

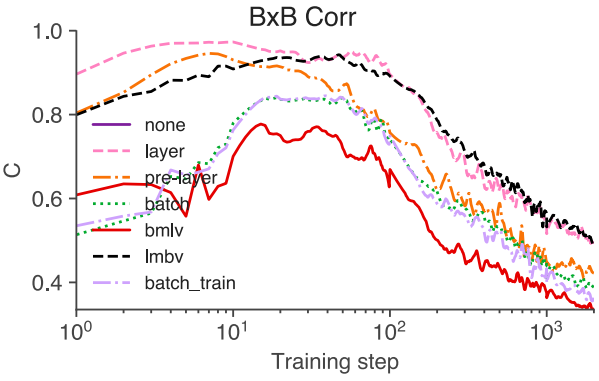

Figure App.4: Correlation between output representations of similar minibatches at the last pre-activation layer through a 26-layer WideResnet ReLU network with skip-connections through the first 2000 steps of training on Cifar10. Even with skip connections, these networks undergo phase shifts where the representations get more correlated early in training and later on are more decorrelated. Note that Batch Norm (and BMLV) are the most decorrelated while PreLayerNorm (and LMBV) is more decorrelated than Layer Norm.

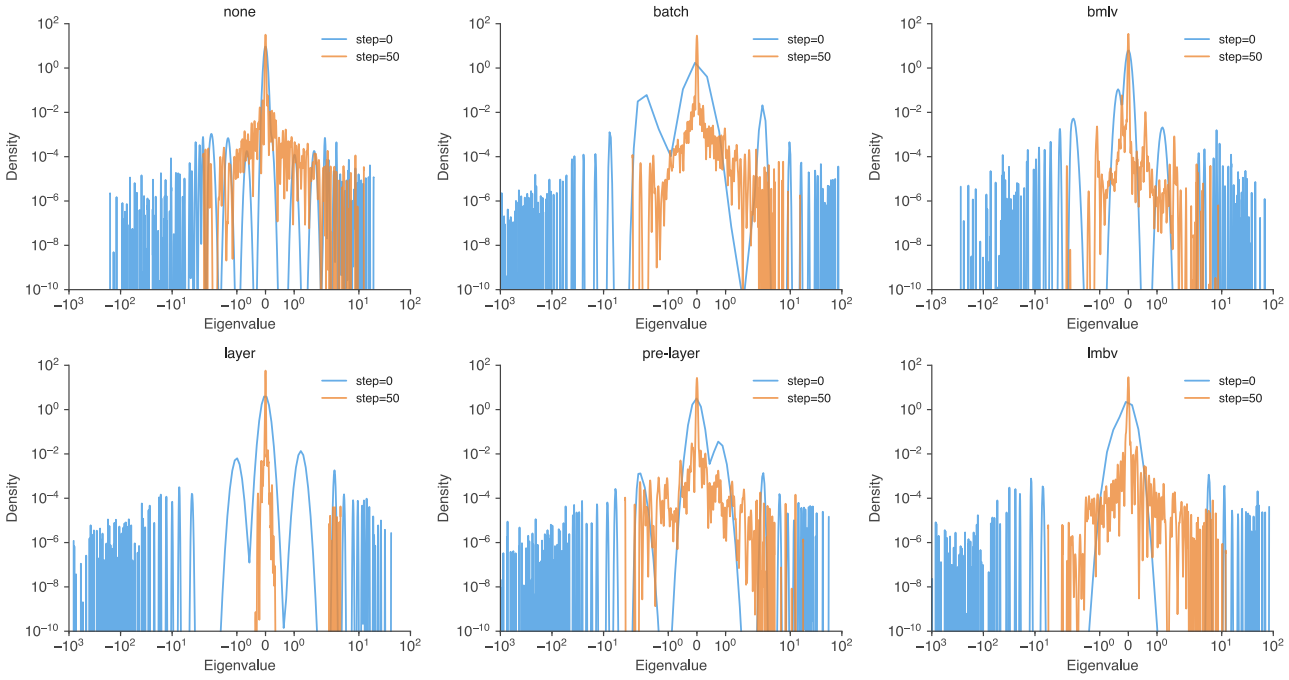

Figure App.5: Density of Eigenvalues of the Hessian (cross-entropy) at initialization and 50 steps through training (we used the same learning rate for all techniques) for a 26-layer WideResnet with ReLU activations and no skip connections. With skip connections, we notice that all the normalization schemes seem to have similar spectra.

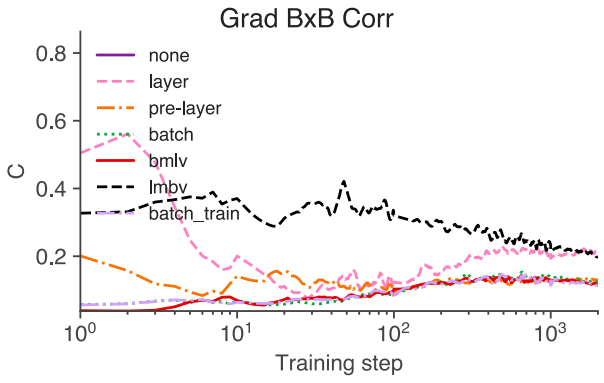

Figure App.6: Correlation between gradients of similar minibatches w.r.t parameters at the last pre-activation layer through a 26-layer WideResnet ReLU network with skip-connections through the first 2000 steps of training on Cifar10. With skip connections, most techniques are able to avoid gradient confusion early in training, except LMBV which does so to a lesser degree.

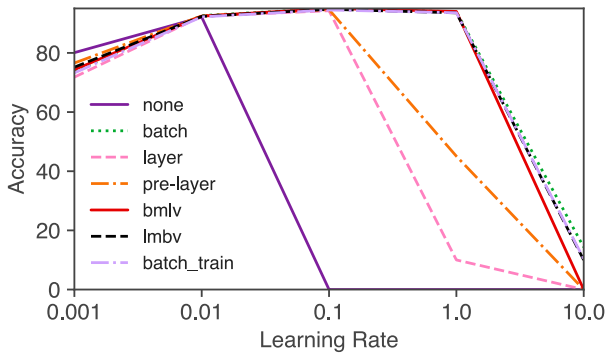

Figure App.7: Effect of using a wide range of learning rates under different normalization techniques. Batch Norm and BMLV allow using larger stable learning rates while PreLayerNorm allows higher learning rates than Layer Norm.

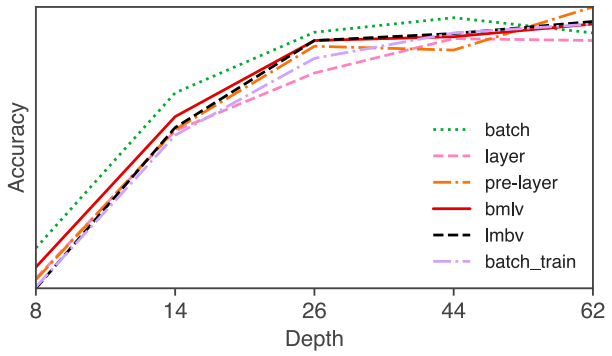

Figure App.8: Effect of increasing depth under different normalization techniques. Most of the normalization techniques used allow training deeper networks, especially with skip connections. Without skip connections, using normalization is imperative for training deeper networks.

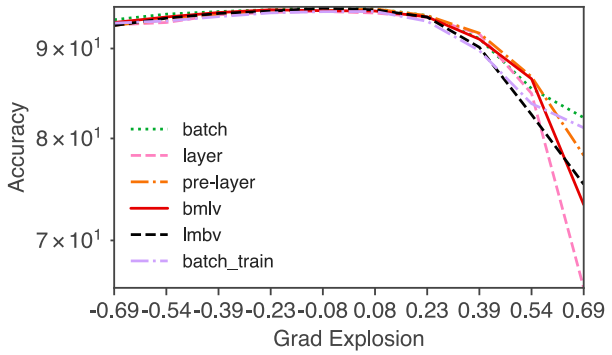

Figure App.9: Induced gradient explosion. This mimics Batch Norm's gradient explosion by introducing a scalar multiplier to the gradients at each layer. Note that the optimal gradient explosion/vanishing just turns out to be $\sim$1 for all the techniques with performance rapidly decreasing when the gradient explosion is high.

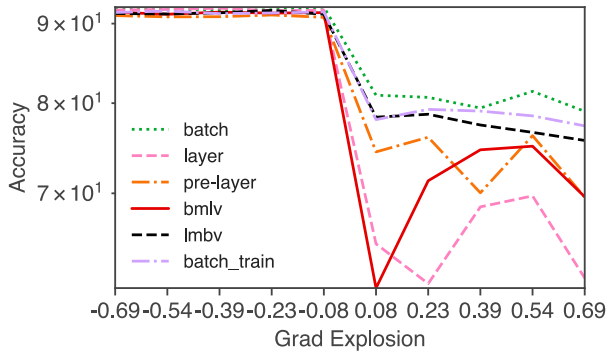

Figure App.10: Induced gradient explosion with a schedule. This mimics Batch Norm's gradient explosion by introducing a scalar multiplier to the gradients at each layer, only for the first 50 steps of training. We arrive at this number by looking at the change in gradient norms of networks with Batch Norm through training. High gradient explosion hurts performance for all the techniques, while with this schedule, networks with slightly vanishing gradients perform consistently with the optimal ($\sim$1)

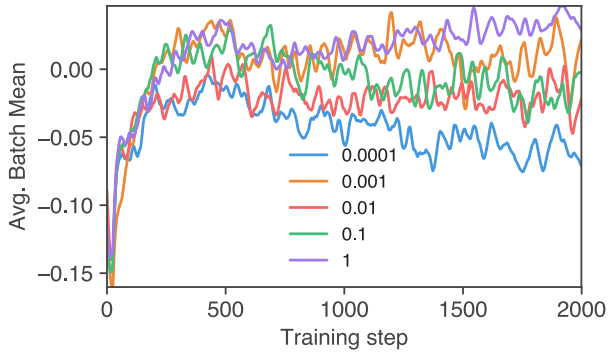

Figure App.11: Effect of using RegNorm on the batch mean of the pre-activations at a random layer in a 26-layer WideResnet. This figure plots the average (across all the channels) batch mean for the first 2000 steps through training against different values of the regularization co-efficient. We observed that larger co-efficients push the mean across the batch to be closer to 0. However, if the co-efficient was too large, it hurt the trainability of the network.