Batch Normalization is a Cause of Adversarial Vulnerability

Angus Galloway [1 2] **Anna Golubeva** [3 4 2] **Thomas Tanay** [5] **Medhat Moussa** [1] **Graham Taylor** [1 2 6]

Abstract

Batch normalization (batch norm) is often used in an attempt to stabilize and accelerate training in deep neural networks. In many cases it indeed decreases the number of parameter updates required to reduce the training error. However, it also reduces robustness to small input perturbations and noise by double-digit percentages, as we show on five standard datasets. Furthermore, substituting weight decay for batch norm is sufficient to nullify the relationship between adversarial vulnerability and the input dimension. Our work is consistent with a mean-field analysis that found that batch norm causes exploding gradients.

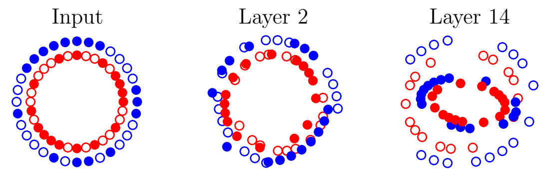


Figure 1. Concentric circles and their representations in a deep linear network with batch norm at initialization. Mini-batch membership is indicated by marker fill and class membership by colour. Each layer is projected to its two principal components. Some samples overlap at Layer 2, and classes are mixed at Layer 14.

1. Introduction

Batch norm is a standard component of modern deep neural networks, and tends to make the training process less sensitive to the choice of hyperparameters in many cases (Ioffe & Szegedy, 2015). While ease of training is desirable for model developers, an important concern among stakeholders is that of model robustness to plausible, previously unseen inputs during deployment.

The adversarial examples phenomenon has exposed unstable predictions across state-of-the-art models (Szegedy et al., 2014). This has led to a variety of methods that aim to improve robustness, but doing so effectively remains an open problem (Athalye et al., 2018; Schott et al., 2019; Hendrycks & Dietterich, 2019; Jacobsen et al., 2019). We believe that a prerequisite to developing methods that increase robustness is an understanding of factors that reduce it.

Approaches for improving robustness often begin with existing neural network architectures—that use batch norm—and patching them against specific attacks, e.g., through inclusion of adversarial examples during training (Szegedy et al., 2014; Goodfellow et al., 2015; Kurakin et al., 2017; Mądry et al., 2018). An implicit assumption is that batch norm itself does not reduce robustness – an assumption that we tested empirically and found to be invalid. In the original work that introduced batch norm, it was suggested that other forms of regularization can be turned down or disabled when using it without decreasing standard test accuracy. Robustness, however, is less forgiving: it is strongly impacted by the disparate mechanisms behind each regularizer.

The observation that adversarial vulnerability can scale with the input dimension (Goodfellow et al., 2015; Simon-Gabriel et al., 2018) highlights the importance of recognizing different regularizers at a deeper level than merely as a way to improve test accuracy. Batch norm was a confounding factor in Simon-Gabriel et al. (2018), making their initialization-time assumptions hold after training. By adding ℓ_2 regularization and removing batch norm, we show that there is no *inherent* relationship between adversarial vulnerability and the input dimension.

2. Batch Normalization

We briefly review how batch norm modifies the hidden layers' pre-activations h of a neural network. We use the notation of Yang et al. (2019), where α is the index for a neuron, l for the layer, and i for a mini-batch of B samples from the dataset; N_l denotes the number of neurons in layer l, W^l is the matrix of weights and b^l is

[1]School of Engineering, University of Guelph [2]Vector Institute for Artificial Intelligence [3]Department of Physics and Astronomy, University of Waterloo [4]Perimeter Institute for Theoretical Physics [5]Huawei Noah's Ark Lab (work done while at University College London) [6]Google Brain. Correspondence to: Angus Galloway, Graham Taylor <gallowaa, gwtaylor@uoguelph.ca>.

Preliminary work. Under review at the ICML 2019 Workshop on Identifying and Understanding Deep Learning Phenomena.

the vector of biases that parametrize layer l. The batch mean is defined as $\mu_\alpha = \frac{1}{B}\sum_i h_{\alpha i}$, and the variance is $\sigma_\alpha^2 = \sqrt{\frac{1}{B}\sum_i (h_{\alpha i} - \mu_\alpha)^2 + c}$, where c is a small constant to prevent division by zero. In the batch norm procedure, the mean μ_α is subtracted from the pre-activation of each neuron $h^l_{\alpha i}$ (consistent with Ioffe & Szegedy (2015)), the result is divided by the standard deviation σ_α, then scaled and shifted by the learned parameters γ_α and β_α, respectively. This is described in Eq. (1), where a per-unit nonlinearity ϕ, e.g., ReLU, is applied after the normalization.

$$h^l_i = W^l \phi(\tilde{h}^{l-1}_i) + b^l, \qquad \tilde{h}^l_{\alpha i} = \gamma_\alpha \frac{h_{\alpha i} - \mu_\alpha}{\sigma_\alpha} + \beta_\alpha \quad (1)$$

Note that this procedure fixes the first and second moments of all neurons α equally. This suppresses the information contained in these moments. Further, batch norm induces a nonlocal batch-wise nonlinearity, such that two mini-batches that differ by only a *single* example will have different representations for *each* example (Yang et al., 2019). This difference is further amplified by stacking batch norm layers. We argue that this information loss and inability to maintain relative distances in the input space reduces adversarial as well as general robustness. Figure 1 shows this effect for a batch-normalized linear network on a 2D variant of the "Adversarial Spheres" dataset (Gilmer et al., 2018).[1]

3. Empirical Result

We first evaluate the robustness (quantified as the drop in test accuracy under input perturbations) of convolutional networks, with and without batch norm, that were trained using standard procedures. The datasets – MNIST, SVHN, CIFAR-10, and ImageNet – were normalized to zero mean and unit variance.

As a white-box adversarial attack we select the basic iterative gradient method (BIM), ℓ_∞- and ℓ_2-norm variants, for its simplicity and ability to degrade performance with little perceptible change to the input (Kurakin et al., 2017). We run BIM for 20 iterations, with $\epsilon_\infty = 0.03$ and a step size of $\epsilon_\infty/20$ for SVHN and CIFAR-10. For BIM-ℓ_2 we set $\epsilon_2 = \epsilon_\infty\sqrt{d}$, where d is the input dimension. We report the test accuracy for additive Gaussian noise of zero mean and variance $1/4$, denoted as "Noise" (Ford et al., 2019), as well as the full CIFAR-10-C common corruption benchmark (Hendrycks & Dietterich, 2019) in Appendix C.

We found these methods were sufficient to demonstrate a considerable disparity in robustness due to batch norm, but this is not intended as a formal security evaluation. All uncertainties are the standard error of the mean.[2]

For the SVHN dataset, models were trained by stochastic gradient descent (SGD) with momentum 0.9 for 50 epochs, with a batch size of 128 and initial learning rate of 0.01, which was dropped by a factor of ten at epochs 25 and 40. Trials were repeated over five random seeds. We show the results of this experiment in Table 1, finding that despite batch norm increasing clean test accuracy by $1.86 \pm 0.05\%$, it reduced test accuracy for additive noise by $5.5 \pm 0.6\%$, for BIM-ℓ_∞ by $17 \pm 1\%$, and for BIM-ℓ_2 by $31 \pm 1\%$.

Table 1. Test accuracies of VGG8 on SVHN.

BN	Clean	Noise	BIM-ℓ_∞	BIM-ℓ_2
✗	92.60 ± 0.04	83.6 ± 0.2	31.3 ± 0.2	39 ± 1
✓	94.46 ± 0.02	78.1 ± 0.6	15 ± 1	8 ± 1

For the CIFAR-10 experiments we trained models with a similar procedure as for SVHN, but with random 32×32 crops using four-pixel padding, and horizontal flips.

In the first experiment, a basic comparison with and without batch norm shown in Table 2, we evaluated the best model in terms of test accuracy over 150 epochs with a fixed learning rate of 0.01. In this case, inclusion of batch norm reduces the clean generalization gap (difference between training and test accuracy) by $1.1 \pm 0.2\%$. For additive noise, test accuracy drops by $6 \pm 1\%$, and for BIM perturbations by $16.9 \pm 0.6\%$ and $12 \pm 1\%$ for ℓ_∞ and ℓ_2 variants, respectively. Very similar results, presented in Table 3, are obtained on a new test set, CIFAR-10.1 `v6` (Recht et al., 2018): batch norm slightly improves the clean test accuracy (by $2.0 \pm 0.3\%$), but leads to a considerable drop in test accuracy for the cases with additive noise and the two BIM variants.

It has been suggested that one of the benefits of batch norm is that it facilitates training with a larger learning rate (Ioffe & Szegedy, 2015; Bjorck et al., 2018). We test this from a robustness perspective in an experiment summarized in Table 4, where the initial learning rate was increased to 0.1 when batch norm was used. We prolonged training for up to 350 epochs, and dropped the learning rate by a factor of ten at epoch 150 and 250 in both cases, which increases clean test accuracy relative to Table 2. The deepest model that was trainable using standard He et al. (2015) initialization without batch norm was VGG13.[3] None of

Table 2. Test accuracies of VGG8 on CIFAR-10.

BN	Clean	Noise	BIM-ℓ_∞	BIM-ℓ_2
✗	87.9 ± 0.1	78.9 ± 0.6	53.8 ± 0.5	44 ± 1
✓	88.7 ± 0.1	73 ± 1	36.9 ± 0.2	31.1 ± 0.4

[1]We add a ReLU nonlinearity when attempting to *learn* the binary classification task posed by Gilmer et al. (2018) in Appendix D, but the activations in the linear case give us pause.

[2]Each experiment has a unique uncertainty, hence the number of decimal places varies.

[3]For which one of ten random seeds failed to achieve better than chance accuracy on the training set, while others performed as expected. We report the mean and standard error for the first

Table 3. Test accuracies of VGG8 on CIFAR-10.1 (`v6`).

BN	Clean	Noise	BIM-ℓ_∞	BIM-ℓ_2
✗	75.3 ± 0.2	66 ± 1	36 ± 1	23 ± 1
✓	77.3 ± 0.2	60 ± 2	20.9 ± 0.7	11.5 ± 0.2

Table 4. VGG models of increasing depth on CIFAR-10, with and without batch norm (BN). See text for differences in hyperparameters compared to Table 2.

Model		Test Accuracy (%)		
L	BN	Clean	Noise	BIM-ℓ_∞
8	✗	89.29 ± 0.09	81.7 ± 0.3	56.3 ± 0.4
8	✓	90.49 ± 0.01	77 ± 1	42.1 ± 0.6
11	✗	90.4 ± 0.1	81.5 ± 0.5	54.8 ± 0.2
11	✓	91.19 ± 0.06	79.3 ± 0.6	45.5 ± 0.4
13	✗	91.74 ± 0.02	77.8 ± 0.7	42.5 ± 0.7
13	✓	93.0 ± 0.1	67 ± 1	31.5 ± 0.3
16	✓	92.8 ± 0.1	66 ± 2	31.2 ± 0.4
19	✓	92.65 ± 0.09	68 ± 2	32.63 ± 0.08

the deeper models with batch norm recover the robustness of the most shallow, or same-depth equivalents without batch norm, nor does the higher learning rate in conjunction with batch norm improve robustness compared to baselines trained for the same number of epochs. Additional results for deeper models on SVHN and CIFAR-10 can be found in Appendix A.

We evaluated models pre-trained on ImageNet from the `torchvision.models` repository using ten iterations of BIM with a step size of 1e-3, or ϵ_∞ = 1e-2.[4] Results are shown in Table 5, where batch norm improves top-5 accuracy on noise in some cases, but consistently reduces it by 8.91% to 10.96% (absolute) for BIM. The trends are the same for top-1 accuracy, only the absolute values were smaller; the degradation varies from 2.92% to 5.32%. Given the discrepancy between noise and BIM for ImageNet, we conduct a black-box transfer analysis in Appendix A.3.

We suspect that the robustness gap due to batch norm for ImageNet is smaller than for other datasets because all models are highly vulnerable by default. We believe this gap could be made larger by training a baseline with greater ℓ_2 regularization (Galloway et al., 2018). For computational reasons, we opt to show this for a simpler dataset in Section 4.

Finally, we explore the role of batch size and depth in Figure 2. Batch norm limits the maximum trainable depth, which *increases* with the batch size, but quickly plateaus as predicted by Theorem 3.10 of Yang et al. (2019). Robustness *decreases* with the batch size for depths that maintain a reasonable test accuracy, at around 25 or fewer layers. This tension between clean accuracy and robustness as a function of the batch size is not observed in unnormalized networks. We show the effect of increasing the number of epochs on these trends in Appendix A.4.

three successful runs for consistency with the other experiments.

[4]https://pytorch.org/docs/stable/torchvision/models.html, `v1.0.0`.

Table 5. Models from `torchvision.models` pre-trained on ImageNet, some with and some without batch norm (BN).

Model		Top 5 Test Accuracy (%)		
Model	BN	Clean	Noise	BIM-ℓ_∞
VGG-11	✗	88.63	49.16	40.03
VGG-11	✓	89.81	49.95	29.07
VGG-13	✗	89.25	52.55	32.50
VGG-13	✓	90.37	52.12	23.59
VGG-16	✗	90.38	60.67	36.47
VGG-16	✓	91.52	65.36	25.83
VGG-19	✗	90.88	64.86	37.88
VGG-19	✓	91.84	68.79	28.70
AlexNet	✗	79.07	41.41	41.31
DenseNet121	✓	91.97	79.85	39.58
ResNet18	✓	88.65	79.62	34.50

4. Vulnerability and Input Dimension

We briefly summarize the "core idea" of Simon-Gabriel et al. (2018) regarding the relation between the adversarial vulnerability and the input dimension d. Consider adversarial perturbations δ of size $\|\delta\|_p \leq \epsilon_p$ to an input x, generated with norm ℓ_p. Adversarial vulnerability is quantified by the variation of the loss function $\mathcal{L}$ under an ϵ-sized attack, which, to first order in ϵ, is given by the dual norm of the loss gradient $\|\nabla_x \mathcal{L}\|_q$, where ${}^1\!/_p + {}^1\!/_q = 1$. Introducing a dimension-independent constant ϵ_∞, the attack size can be expressed as $\epsilon_p = \epsilon_\infty d^{1/p}$, such that

$$\epsilon_p \|\nabla_x \mathcal{L}\|_q \propto \epsilon_p \, d^{1/q} |\nabla_x \mathcal{L}| \propto d |\nabla_x \mathcal{L}|. \tag{2}$$

Under the commonly used He et al. (2015) initialization scheme, weights are initialized with mean zero and variance $1/d$. Considering the simplest case of a linear single-unit network, Simon-Gabriel et al. conjecture that adversarial vulnerability scales as $\sim \sqrt{d}$ at initialization time. They also show in experiments that independence between vulnerability and the input dimension can be approximately recovered through adversarial training by projected gradient descent (PGD) (Mądry et al., 2018), with a modest trade-off of clean accuracy. We show that this can be achieved by simpler means and with little to no trade-off through ℓ_2 weight decay, where the regularization constant λ corrects the scaling of the loss in higher dimensions.

We increase the MNIST image width ($\sqrt{d}$) from 28 to 56,

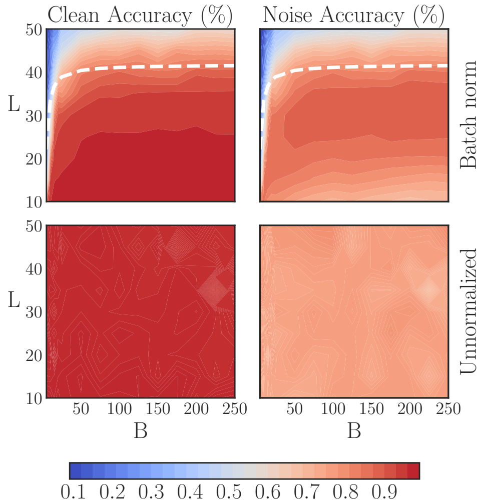


Figure 2. We repeat the experiment of Yang et al. (2019) by training fully-connected nets of depth L and constant-width ReLU layers for ten epochs by SGD, and learning rate $\eta = 10^{-5}B$ for batch size B on MNIST. The batch norm parameters γ and β were left as default, momentum was disabled, and $c = 10^{-3}$. The dashed line is the theoretical maximum trainable depth as a function of batch size. Trials averaged over three random seeds.

Table 6. Evaluating the robustness of a MLP with and without batch norm. We observe a $61 \pm 1\%$ reduction in test accuracy due to batch norm for $\sqrt{d} = 84$ compared to $\sqrt{d} = 28$.

Model		Test Accuracy (%)		
$\sqrt{d}$	BN	Clean	Noise	$\epsilon = 0.1$
28	✗	97.95 ± 0.08	93.0 ± 0.4	66.7 ± 0.9
	✓	97.88 ± 0.09	76.6 ± 0.7	22.9 ± 0.7
56	✗	98.19 ± 0.04	93.8 ± 0.1	53.2 ± 0.7
	✓	98.22 ± 0.02	79.3 ± 0.6	8.6 ± 0.8
84	✗	98.27 ± 0.04	94.3 ± 0.1	47.6 ± 0.8
	✓	98.28 ± 0.05	80.5 ± 0.6	6.1 ± 0.5

Table 7. Evaluating the robustness of a MLP with ℓ_2 weight decay (same λ as for linear model, see Table 11 of Appendix B). Adding batch norm degrades all accuracies.

Model		Test Accuracy (%)		
$\sqrt{d}$	BN	Clean	Noise	$\epsilon = 0.1$
56	✗	97.62 ± 0.06	95.93 ± 0.06	87.9 ± 0.2
	✓	96.23 ± 0.03	90.22 ± 0.18	66.2 ± 0.8
84	✗	96.99 ± 0.05	95.69 ± 0.09	87.9 ± 0.1
	✓	93.30 ± 0.09	87.72 ± 0.11	65.1 ± 0.5

84, and 112 pixels. We confirm that without regularization the increase in the loss $\mathcal{L}$, which was predicted to grow like $\sqrt{d}$ by Thm. 4 of Simon-Gabriel et al., attains values roughly between those obtained empirically for $\epsilon = 0.05$ and $\epsilon = 0.1$ at all image widths. Training with ℓ_2 weight decay, however, we obtain adversarial test accuracy ratios of 0.98 ± 0.01, 0.96 ± 0.04, and 1.00 ± 0.03 and clean accuracy ratios of 0.999 ± 0.002, 0.996 ± 0.003, and 0.987 ± 0.004 for $\sqrt{d}$ of 56, 84, and 112 respectively, relative to the original $\sqrt{d} = 28$ dataset. A theoretical explanation for this and more detailed results are provided in Appendix B.

Next, we repeat this experiment with a two-hidden-layer ReLU MLP, and optionally use one batch-norm hidden layer. To evaluate robustness, 100 iterations of BIM-ℓ_∞ were used with a step size of 1e-3, and $\epsilon_\infty = 0.1$. We also report test accuracy with additive Gaussian noise of zero mean and unit variance, the same first two moments as the clean images.[5]

Despite a difference in clean accuracy of only $0.08 \pm 0.05\%$, Table 6 shows that for the original image resolution, batch norm reduced accuracy for noise by $16.4 \pm 0.4\%$, and for BIM-ℓ_∞ by $43.8 \pm 0.5\%$. Robustness keeps decreasing as the image size increases, with the batch-normalized network having $\sim 40\%$ less robustness to BIM and $13 - 16\%$ less to noise at all sizes.

[5]We first apply the noise to the original 28×28 pixel images, then resize them to preserve the appearance of the noise.

We then apply the ℓ_2 regularization constants tuned for the respective input dimensions on the linear model to the MLP with no further adjustments. Table 7 shows that by adding sufficient ℓ_2 regularization ($\lambda = 0.01$) to recover the original ($\sqrt{d} = 28$, no BN) accuracy for BIM of $\approx 66\%$ when using batch norm, we increase the test error by $1.69 \pm 0.01\%$, which is substantial on MNIST. Furthermore, using the same regularization constant without batch norm increases clean test accuracy by $1.39 \pm 0.04\%$, and for the case of BIM-ℓ_∞ perturbation by $21.7 \pm 0.4\%$.

In all cases, using batch norm reduced test accuracy under attacks by a large margin, while weight decay increases it. Following the guidance in the original work (Ioffe & Szegedy, 2015) to the extreme ($\lambda = 0$): to *reduce* weight decay when using batch norm, we degrade accuracy for the $\epsilon_\infty = 0.1$ perturbations by $79.3 \pm 0.3\%$ for $\sqrt{d} = 56$, and $81.2 \pm 0.2\%$ for $\sqrt{d} = 84$.

5. Conclusion

We found that there is no free lunch with batch norm: the accelerated training properties and occasionally higher clean test accuracy come at the cost of robustness, both to additive noise and for adversarial perturbations. We have shown that there is no inherent relationship between the input dimension and vulnerability. Our results highlight the importance of identifying the disparate mechanisms of regularization techniques, especially when concerned about robustness.

Acknowledgements

The authors wish to acknowledge the financial support of NSERC, CFI, CIFAR and EPSRC. We also acknowledge hardware support from NVIDIA and Compute Canada. Research at the Perimeter Institute is supported by Industry Canada and the province of Ontario through the Ministry of Research & Innovation. We thank Thorsteinn Jonsson for helpful discussions; Colin Brennan, Terrance DeVries and Jörn-Henrik Jacobsen for technical suggestions; Justin Gilmer for suggesting the common corruption benchmark; Maeve Kennedy, Vithursan Thangarasa, Katya Kudashkina, and Boris Knyazev for comments and proofreading.

References

Athalye, A., Carlini, N., and Wagner, D. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *International Conference on Machine Learning*, pp. 274–283, 2018.

Bjorck, N., Gomes, C. P., Selman, B., and Weinberger, K. Q. Understanding Batch Normalization. In *Advances in Neural Information Processing Systems 31*, pp. 7705–7716. 2018.

Brendel, W. and Bethge, M. Approximating CNNs with Bag-of-local-Features models works surprisingly well on ImageNet. In *International Conference on Learning Representations*, 2019.

Ford, N., Gilmer, J., and Cubuk, E. D. Adversarial Examples Are a Natural Consequence of Test Error in Noise. 2019.

Galloway, A., Tanay, T., and Taylor, G. W. Adversarial Training Versus Weight Decay. *arXiv preprint arXiv:1804.03308*, 2018.

Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., and Brendel, W. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019.

Gilmer, J., Metz, L., Faghri, F., Schoenholz, S., Raghu, M., Wattenberg, M., and Goodfellow, I. Adversarial Spheres. In *International Conference on Learning Representations Workshop Track*, 2018.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *International Conference on Computer Vision*, pp. 1026–1034. IEEE Computer Society, 2015.

Hendrycks, D. and Dietterich, T. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. In *International Conference on Learning Representations*, 2019.

Hoffer, E., Hubara, I., and Soudry, D. Train longer, generalize better: Closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems 30*, pp. 1731–1741. 2017.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.

Jacobsen, J.-H., Behrmann, J., Carlini, N., Tramèr, F., and Papernot, N. Exploiting Excessive Invariance caused by Norm-Bounded Adversarial Robustness. *Safe Machine Learning workshop at ICLR*, 2019.

Kurakin, A., Goodfellow, I. J., and Bengio, S. Adversarial Machine Learning at Scale. *International Conference on Learning Representations*, 2017.

Mądry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*, 2018.

Rauber, J., Brendel, W., and Bethge, M. Foolbox: A Python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017.

Recht, B., Roelofs, R., Schmidt, L., and Shankar, V. Do CIFAR-10 Classifiers Generalize to CIFAR-10? *arXiv:1806.00451*, 2018.

Schmidt, L., Santurkar, S., Tsipras, D., Talwar, K., and Mądry, A. Adversarially Robust Generalization Requires More Data. In *Advances in Neural Information Processing Systems 31*, pp. 5014–5026. 2018.

Schott, L., Rauber, J., Bethge, M., and Brendel, W. Towards the first adversarially robust neural network model on MNIST. In *International Conference on Learning Representations*, 2019.

Sculley, D., Snoek, J., Wiltschko, A., and Rahimi, A. Winner's Curse? On Pace, Progress, and Empirical Rigor. In *International Conference on Learning Representations, Workshop*, 2018.

Simon-Gabriel, C.-J., Ollivier, Y., Bottou, L., Schölkopf, B., and Lopez-Paz, D. Adversarial Vulnerability of Neural Networks Increases With Input Dimension. *arXiv:1802.01421*, 2018.

Soudry, D., Hoffer, E., Nacson, M. S., and Srebro, N. The Implicit Bias of Gradient Descent on Separable Data. In *International Conference on Learning Representations*, 2018.

Su, D., Zhang, H., Chen, H., Yi, J., Chen, P.-Y., and Gao, Y. Is Robustness the Cost of Accuracy? – A Comprehensive Study on the Robustness of 18 Deep Image Classification Models. In *Computer Vision – ECCV 2018*, pp. 644–661, 2018.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

Tanay, T. and Griffin, L. D. A Boundary Tilting Persepective on the Phenomenon of Adversarial Examples. *arXiv:1608.07690*, 2016.

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Mądry, A. Robustness May Be at Odds with Accuracy. In *International Conference on Learning Representations*, 2019.

van Laarhoven, T. L2 Regularization versus Batch and Weight Normalization. *arXiv:1706.05350*, 2017.

Yang, G., Pennington, J., Rao, V., Sohl-Dickstein, J., and Schoenholz, S. S. A Mean Field Theory of Batch Normalization. In *International Conference on Learning Representations*, 2019.

Zhang, G., Wang, C., Xu, B., and Grosse, R. Three Mechanisms of Weight Decay Regularization. In *International Conference on Learning Representations*, 2019a.

Zhang, H., Dauphin, Y. N., and Ma, T. Residual Learning Without Normalization via Better Initialization. In *International Conference on Learning Representations*, 2019b.

A. Appendix to Empirical Results

This section contains supplementary explanations and results to those of Section 3.

A.1. Why the VGG Architecture?

For SVHN and CIFAR-10 experiments, we selected the VGG family of models as a simple yet contemporary convolutional architecture whose development occurred independent of batch norm. This makes it suitable for a causal intervention, given that we want to study the effect of batch norm itself, and not batch norm + other architectural innovations + hyperparameter tuning. State-of-the-art architectures such as Inception, and ResNet whose development is more intimately linked with batch norm may be less suitable for this kind of analysis. The superior standard test accuracy of these models is somewhat moot given a trade-off between standard test accuracy and robustness, demonstrated in this work and elsewhere (Tanay & Griffin, 2016; Galloway et al., 2018; Su et al., 2018; Tsipras et al., 2019). Aside from these reasons, and provision of pre-trained variants on ImageNet with and without batch norm in `torchvision.models` for ease of reproducibility, this choice of architecture is arbitrary.

A.2. Additional SVHN and CIFAR-10 Results for Deeper Models

Our first attempt to train VGG models on SVHN with more than 8 layers failed, therefore for a fair comparison we report the robustness of the deeper models that were only trainable by using batch norm in Table 8. None of these models obtained much better robustness in terms of BIM-ℓ_2, although they did better for BIM-ℓ_∞.

Table 8. VGG variants on SVHN with batch norm.

	Test Accuracy (%)			
L	Clean	Noise	BIM-ℓ_∞	BIM-ℓ_2
11	95.31 ± 0.03	80.5 ± 1	29.6 ± 2	14.7 ± 2
13	95.88 ± 0.05	77.2 ± 7	30.9 ± 3	13.7 ± 5
16	94.59 ± 0.05	78.1 ± 4	29.3 ± 3	9.0 ± 2
19	95.1 ± 0.3	78 ± 1	29 ± 2	8.8 ± 4

Fixup initialization was recently proposed to reduce the use of normalization layers in deep residual networks (Zhang et al., 2019b). As a natural test we compare a WideResNet (28 layers, width factor 10) with Fixup versus the default architecture with batch norm. Note that the Fixup variant still contains one batch norm layer before the classification layer, but the number of batch norm layers is still greatly reduced.[6]

Table 9. Accuracies of WideResNet–28–10 on CIFAR-10 and CIFAR-10.1 (`v6`).

	CIFAR-10				CIFAR-10.1	
Model	Clean	Noise	BIM-ℓ_∞	BIM-ℓ_2	Clean	Noise
Fixup	94.6 ± 0.1	69.1 ± 1.1	13.1 ± 0.5	27.5 ± 0.5	87.5 ± 0.3	67.8 ± 0.9
BN	95.9 ± 0.1	57.6 ± 1.5	14.5 ± 0.5	25.9 ± 0.9	89.6 ± 0.2	58.3 ± 1.2

We train WideResNets (WRN) with five unique seeds and show their test accuracies in Table 9. Consistent with Recht et al. (2018), higher clean test accuracy on CIFAR-10, i.e. obtained by the WRN compared to VGG, translated to higher clean accuracy on CIFAR-10.1. However, these gains were wiped out by moderate Gaussian noise. VGG8 dramatically outperforms both WideResNet variants subject to noise, achieving 78.9 ± 0.6 vs. 69.1 ± 1.1. Unlike for VGG8, the WRN showed little generalization gap between noisy CIFAR-10 and 10.1 variants: 69.1 ± 1.1 is reasonably compatible with 67.8 ± 0.9, and 57.6 ± 1.5 with 58.3 ± 1.2. The Fixup variant obtains an $11.6 \pm 1.9\%$ improvement in accuracy for noise on CIFAR-10, and a $9.5 \pm 1.5\%$ improvement for CIFAR-10.1. Performance is comparable for the adversarial perturbations,

[6] We used the implementation from https://github.com/valilenk/fixup, but stopped training at 150 epochs for consistency with the VGG8 experiment. Both models had already fit the training set by this point.

with a slight improvement in terms of BIM-ℓ_2. We compared our BIM-ℓ_∞ implementation with that of Foolbox (Rauber et al., 2017) on a 2048 sample subset of the test set using a step size of $\epsilon/15$ for 20 iterations and found that batch norm reduced accuracy by $5.9 \pm 0.2\%$; we opt to report the null result obtained by our implementation on the full test set for consistency with the other tables. We believe our work serves as a compelling motivation for Fixup and other techniques that aim to reduce usage of batch normalization. The role of skip-connections should be isolated in future work.

A.3. ImageNet Black-box Transferability Analysis

Table 10. ImageNet validation accuracy for adversarial examples transfered between VGG variants of various depths, indicated by number, with and without batch norm ("✓", "✗"). All adversarial examples were crafted with BIM-ℓ_∞ using 10 steps and a step size of 5e-3, which is higher than for the white-box analysis to improve transferability. The BIM objective was simply misclassification, i.e., it was not a targeted attack. For efficiency reasons, we select 2048 samples from the validation set. Values along the diagonal in first two columns for Source = Target indicate white-box accuracy.

			Target							
			11		13		16		19	
Acc. Type	Source		✗	✓	✗	✓	✗	✓	✗	✓
Top 1	11	✗	1.2	42.4	37.8	42.9	43.8	49.6	47.9	53.8
		✓	58.8	0.3	58.2	45.0	61.6	54.1	64.4	58.7
Top 5	11	✗	11.9	80.4	75.9	80.9	80.3	83.3	81.6	85.1
		✓	87.9	6.8	86.7	83.7	89.0	85.7	90.4	88.1

The discrepancy between the results in additive noise and for white-box BIM perturbations for ImageNet in Section 3 raises a natural question: Is *gradient masking* a factor influencing the success of the white-box results on ImageNet? No, consistent with the white-box results, when the target is unnormalized but the source is, top 1 accuracy is $10.5\% - 16.4\%$ higher, while top 5 accuracy is $5.3\% - 7.5\%$ higher, than vice versa. This can be observed in Table 10 by comparing the diagonals from lower left to upper right. When targeting an unnormalized model, we reduce top 1 accuracy by $16.5\% - 20.4\%$ using a source that is also unnormalized, compared to a difference of only $2.1\% - 4.9\%$ by matching batch normalized networks. This suggests that the features used by unnormalized networks are more stable than those of batch normalized networks.

Unfortunately, the pre-trained ImageNet models provided by the PyTorch developers do not include hyperparameter settings or other training details. However, we believe that this speaks to the generality of the results, i.e., that they are not sensitive to hyperparameters.

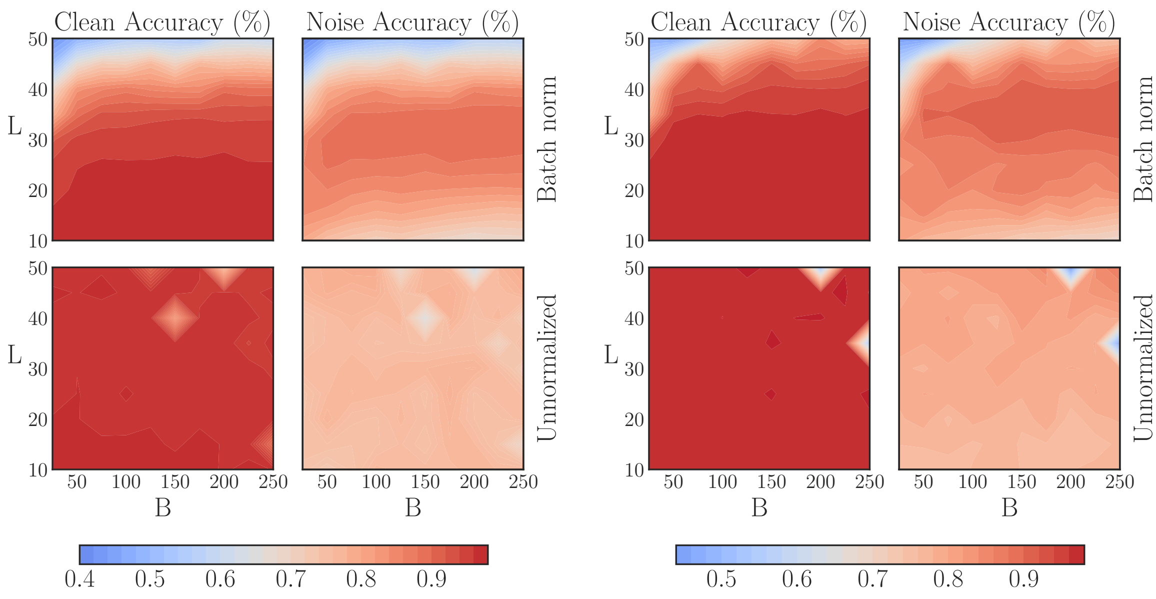
A.4. Batch Norm Limits Maximum Trainable Depth and Robustness

In Figure 3 we show that batch norm not only limits the maximum trainable depth, but robustness decreases with the batch size for depths that maintain test accuracy, at around 25 or fewer layers (in Figure 3(a)). Both clean accuracy and robustness showed little to no relationship with depth nor batch size in unnormalized networks. A few outliers are observed for unnormalized networks at large depths and batch size, which could be due to the reduced number of parameter update steps that result from a higher batch size and fixed number of epochs (Hoffer et al., 2017).

Note that in Figure 3(a) the bottom row—without batch norm—appears lighter than the equivalent plot above, with batch norm, indicating that unnormalized networks obtain less absolute peak accuracy than the batch normalized network. Given that the unnormalized networks take longer to converge, we prolong training for 40 total epochs. When they do converge, we see more configurations that achieve higher clean test accuracy than batch normalized networks in Figure 3(b). Furthermore, good robustness can be experienced simultaneously with good clean test accuracy in unnormalized networks, whereas the regimes of good clean accuracy and robustness are still non-overlapping in Figure 3(b).

B. Weight Decay and Input Dimension

Consider a logistic regression model with input $x \in \mathbb{R}^d$, labels $y \in \{\pm 1\}$, parameterized by weights w, and bias b. Predictions are defined by $o = w^\top x + b$ and the model can be optimized by stochastic gradient descent (SGD) on the



(a) (b)

Figure 3. We repeat the experiment of Yang et al. (2019) by training fully-connected models of depth L and constant width (N_l=384) with ReLU units by SGD, and learning rate $\eta = 10^{-5}B$ for batch size B on MNIST. We train for 10 and 40 epochs in (a) and (b) respectively. The batch norm parameters γ and β were left as default, momentum disabled, and c = 1e-3. Each coordinate is first averaged over three seeds. Diamond shaped artefacts for unnormalized case indicate one of three seeds failed to train – note that we show an equivalent version of (a) with these outliers removed and additional batch sizes from 5–20 in Figure 2. Best viewed in colour.

sigmoid cross entropy loss, which reduces to SGD on (3), where ζ is the softplus loss $\zeta(z) = \log(1 + e^{-z})$:

$$\mathbb{E}_{x,y\sim p_{\text{data}}}\, \zeta(y(w^\top x + b)). \tag{3}$$

We note that $w^\top x + b$ is a *scaled*, signed distance between x and the classification boundary defined by our model. If we define $d(x)$ as the signed Euclidean distance between x and the boundary, then we have: $w^\top x + b = \|w\|_2\, d(x)$. Hence, minimizing (3) is equivalent to minimizing

$$\mathbb{E}_{x,y\sim p_{\text{data}}}\, \zeta(\|w\|_2 \times y\, d(x)). \tag{4}$$

We define the *scaled softplus loss* as

$$\zeta_{\|w\|_2}(z) := \zeta(\|w\|_2 \times z) \tag{5}$$

and note that adding a ℓ_2 regularization term in (4), resulting in (6), can be understood as a way of controlling the scaling of the softplus function:

$$\mathbb{E}_{x,y\sim p_{\text{data}}}\, \zeta_{\|w\|_2}(y\, d(x)) + \lambda\|w\|_2 \tag{6}$$

In Figures 4(a)-4(c), we develop intuition for the different quantities contained in (3) with respect to a typical binary classification problem, while Figures 4(d)-4(f) depict the effect of the regularization parameter λ on the scaling of the loss function.

To test this theory empirically we study a single linear layer on variants of MNIST of increasing input dimension, where the "core idea" from (Simon-Gabriel et al., 2018) is exact. Clearly, this model is too simple to obtain competitive test accuracy, but this is a helpful first step that will be subsequently extended to ReLU networks. The model was trained by SGD for 50 epochs with a constant learning rate of 1e-2 and a batch size of 128. In Table 11 we show that increasing the input dimension by resizing MNIST from 28×28 to various resolutions with `PIL.Image.NEAREST` interpolation increases adversarial vulnerability in terms of accuracy and loss. Furthermore, the "adversarial damage", which is predicted to grow like $\sqrt{d}$ by

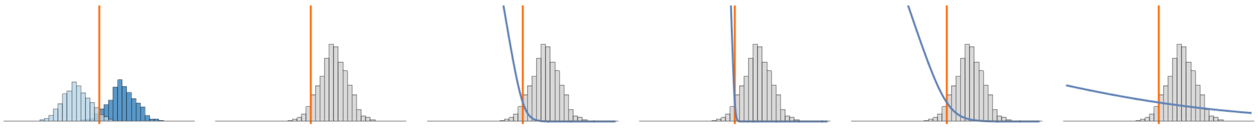


Figure 4. (a) For a given weight vector w and bias b, the values of $w^\top x + b$ over the training set typically follow a bimodal distribution (corresponding to the two classes) centered on the classification boundary. (b) Multiplying by the label y allows us to distinguish the correctly classified data in the positive region from misclassified data in the negative region. (c) We can then attribute a penalty to each training point by applying the softplus loss to $y(w^\top x + b)$. (d) For a small regularization parameter (large $\|w\|_2$), the misclassified data is penalized linearly while the correctly classified data is not penalized. (e) A medium regularization parameter (medium $\|w\|_2$) corresponds to smoothly blending the margin. (f) For a large regularization parameter (small $\|w\|_2$), all data points are penalized almost linearly.

Theorem 4 of Simon-Gabriel et al. (2018), falls in between that obtained empirically for $\epsilon = 0.05$ and $\epsilon = 0.1$ for all image widths except for 112, which experiences slightly more damage than anticipated.

Simon-Gabriel et al. (2018) note that independence between vulnerability and input dimension can be recovered through adversarial example augmented training by projected gradient descent (PGD), with a small trade-off in terms of standard test accuracy. We find that the same can be achieved through a much simpler approach: ℓ_2 weight decay, with λ chosen to correct for the loss scaling. This way we recover input dimension invariant vulnerability with little degradation of test accuracy, e.g., see the $\epsilon = 0.1$ accuracy ratio of 1.00 ± 0.03 with ℓ_2 for $\sqrt{d} = 112$ in Table 11 compared to 0.10 ± 0.09 without.

Compared to PGD training, weight decay regularization i) does not have an arbitrary ϵ hyperparameter that ignores inter-sample distances, ii) does not prolong training by a multiplicative factor given by the number of steps in the inner loop, and 3) is less attack-specific. Thus, we do not use adversarially augmented training because we wish to convey a notion of robustness to unseen attacks and common corruptions. Furthermore, enforcing robustness to ϵ-perturbations may increase vulnerability to *invariance-based* examples, where semantic changes are made to the input thus changing the Oracle label, but not the classifier's prediction (Jacobsen et al., 2019). Our models trained with weight decay obtained 12% higher accuracy (86 vs. 74 correct) compared to batch norm on a small sample of 100 ℓ_∞ invariance-based MNIST examples.[7] We make primary use of traditional ℓ_p perturbations as they are well studied in the literature and straightforward to compute, but solely defending against these is not the end goal.

A more detailed comparison between adversarial training and weight decay can be found in Galloway et al. (2018). The scaling of the loss function mechanism of weight decay is complementary to other mechanisms identified in the literature recently, for instance that it also increases the effective learning rate (van Laarhoven, 2017; Zhang et al., 2019a). Our results are consistent with these works in that weight decay reduces the generalization gap, even in batch-normalized networks where it is presumed to have no effect. Given that batch norm is not typically used on the last layer, the loss scaling mechanism persists in this setting although to a lesser degree.

C. Common Corruption Robustness

We evaluated robustness on the common corruptions and perturbations benchmarks (Hendrycks & Dietterich, 2019). Common corruptions are nineteen types of real world effects that can be grouped into four categories: "noise", "blur", "weather", and "digital". Each corruption has five "severity" or intensity levels. These are applied to the test sets of CIFAR-10 and ImageNet, denoted CIFAR-10-C and ImageNet-C respectively. When reporting the mean corruption error (mCE) we average over intensity levels for each corruption, then over all corruptions. We outline the results for two VGG variants and a WideResNet on CIFAR-10-C, trained from scratch independently over three and five random seeds respectively. The most important results are also summarized in Table 13.

For VGG8 batch norm increased the error rate for all noise variants, at every intensity level. The mean generalization gaps for noise were: Gaussian—$9.2 \pm 1.9\%$, Impulse—$7.5 \pm 0.8\%$, Shot—$5.6 \pm 1.6\%$, and Speckle—$4.5 \pm 1.6\%$. The next most impactful corruptions were: Contrast—$4.4 \pm 1.3\%$, Spatter—$2.4 \pm 0.7\%$, JPEG—$2.0 \pm 0.4\%$, and Pixelate—$1.3 \pm 0.5\%$.

[7] Invariance based adversarial examples downloaded from https://github.com/ftramer/Excessive-Invariance.

Table 11. Mitigating the effect of the input dimension on adversarial vulnerability by correcting the margin enforced by the loss function. Regularization constant λ is for ℓ_2 weight decay. Consistent with Simon-Gabriel et al. (2018), we use ϵ-FGSM perturbations, the optimal ℓ_∞ attack for a linear model. Values in rows with $\sqrt{d} > 28$ are a ratio of entry (accuracy or loss) for the $\sqrt{d} = 28$ baseline. Pred. is the predicted adversarial damage using Thm. 4 of Simon-Gabriel et al., i.e., the increase in $\mathcal{L}$ due to the ϵ_∞ perturbation.

Model		Test Accuracy (%) (See Caption)		Loss (See Caption)			
$\sqrt{d}$	λ	Clean	$\epsilon = 0.1$	Clean	$\epsilon = 0.05$	$\epsilon = 0.1$	Pred.
28	–	$92.4 \pm 0.1\%$	$53.9 \pm 0.3\%$	0.268 ± 0.001	0.646 ± 0.001	1.410 ± 0.004	-
56	–	1.001 ± 0.001	0.33 ± 0.03	1.011 ± 0.007	1.802 ± 0.006	2.449 ± 0.009	2
56	0.01	0.999 ± 0.002	0.98 ± 0.01	1.010 ± 0.007	1.010 ± 0.006	1.01 ± 0.01	-
84	–	0.998 ± 0.002	0.10 ± 0.09	1.06 ± 0.01	2.84 ± 0.02	4.15 ± 0.02	3
84	0.0225	0.996 ± 0.003	0.96 ± 0.04	1.05 ± 0.02	1.06 ± 0.03	1.06 ± 0.03	-
112	–	0.992 ± 0.004	0.1 ± 0.2	1.18 ± 0.03	4.15 ± 0.02	5.96 ± 0.02	4
112	0.05	0.987 ± 0.004	1.00 ± 0.03	1.14 ± 0.04	1.08 ± 0.03	1.04 ± 0.03	-

Table 12. Three-layer ReLU MLP, with and without batch norm (BN), trained for 50 epochs and repeated over five (5) random seeds. Values in rows with $\sqrt{d} > 28$ are a ratio of entry (accuracy or loss) for the $\sqrt{d} = 28$ baseline. There is a considerable increase in adversarial vulnerability in terms of the loss, or similarly, a degradation of robustness in terms of accuracy, due to batch norm. The discrepancy for BIM-ℓ_∞ with $\epsilon = 0.1$ for $\sqrt{d} = 84$ with batch norm represents a $61 \pm 1\%$ degradation in absolute accuracy compared to the baseline.

Model		Test Accuracy (%) (See Caption)			Loss (See Caption)		
$\sqrt{d}$	BN	Clean	Noise	$\epsilon = 0.1$	Clean	$\epsilon = 0.05$	$\epsilon = 0.1$
28	✗	97.95 ± 0.08	93.0 ± 0.4	66.7 ± 0.9	0.0669 ± 0.0008	0.285 ± 0.003	1.06 ± 0.02
28	✓	0.9992 ± 0.0012	0.82 ± 0.01	0.34 ± 0.03	1.06 ± 0.04	2.20 ± 0.03	3.18 ± 0.03
56	✗	1.0025 ± 0.0009	1.009 ± 0.004	0.80 ± 0.02	0.87 ± 0.02	1.27 ± 0.01	1.68 ± 0.03
56	✓	1.0027 ± 0.0008	0.853 ± 0.008	0.13 ± 0.09	0.91 ± 0.03	3.48 ± 0.02	5.83 ± 0.03
84	✗	1.0033 ± 0.0009	1.015 ± 0.004	0.71 ± 0.02	0.86 ± 0.02	1.48 ± 0.02	2.15 ± 0.03
84	✓	1.0033 ± 0.0010	0.865 ± 0.009	0.09 ± 0.08	0.88 ± 0.02	4.34 ± 0.02	7.34 ± 0.02

Table 13. Robustness of three modern convolutional neural network architectures with and without batch norm on the `CIFAR-10-C` common corruptions benchmark (Hendrycks & Dietterich, 2019). We use “F” to denote Fixup (Zhang et al., 2019b) as this variant still had one batch norm layer. Values were averaged over five intensity levels for each corruption. We report the top five of nineteen corruptions by magnitude of the accuracy gap due to batch norm; see the text for more detail on the corruptions omitted here.

Model		Test Accuracy (%)					
Variant	BN	Clean	Gaussian	Impulse	Shot	Speckle	Contrast
VGG8	✗	87.9 ± 0.1	$\mathbf{65.6 \pm 1.2}$	$\mathbf{58.8 \pm 0.8}$	$\mathbf{71.0 \pm 1.2}$	$\mathbf{70.8 \pm 1.2}$	$\mathbf{59.3 \pm 0.8}$
	✓	88.7 ± 0.1	56.4 ± 1.5	51.2 ± 0.1	65.4 ± 1.1	66.3 ± 1.1	54.9 ± 1.0
VGG13	✗	91.74 ± 0.02	$\mathbf{64.5 \pm 0.8}$	$\mathbf{63.3 \pm 0.3}$	$\mathbf{70.9 \pm 0.4}$	$\mathbf{71.5 \pm 0.5}$	65.3 ± 0.6
	✓	93.0 ± 0.1	43.6 ± 1.2	49.7 ± 0.5	56.8 ± 0.9	60.4 ± 0.7	$\mathbf{67.7 \pm 0.5}$
WRN–28–10	F	94.6 ± 0.1	$\mathbf{63.3 \pm 0.9}$	$\mathbf{66.7 \pm 0.9}$	$\mathbf{71.7 \pm 0.7}$	$\mathbf{73.5 \pm 0.6}$	81.2 ± 0.7
	✓	95.9 ± 0.1	51.2 ± 2.7	56.0 ± 2.7	63.0 ± 2.5	66.6 ± 2.5	$\mathbf{86.0 \pm 0.9}$

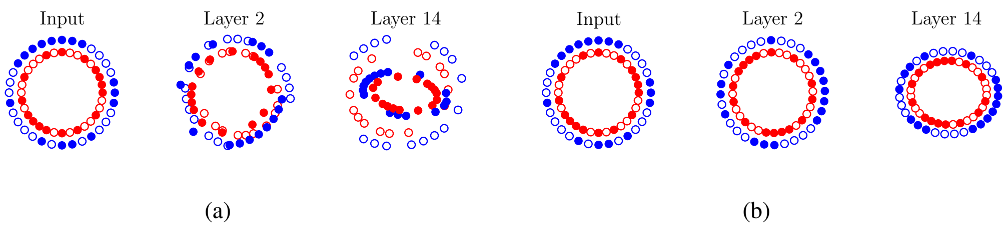


Figure 5. Concentric circles and their representations in a deep linear network (a) with batch norm and (b) without batch norm, at initialization. Mini-batch membership is indicated by marker fill and class membership by colour. Each layer is projected to its two principal components. In (b) we scale both components by a factor of 100 as the dynamic range decreases with depth under default initialization. We observe in (a) that some samples are already overlapping at Layer 2, and classes are mixed at Layer 14.

Results for the remaining corruptions were a coin toss as to whether batch norm improved or degraded robustness, as the random error was in the same ballpark as the difference being measured. These were: Weather—Brightness, Frost, Snow, and Saturate; Blur—Defocus, Gaussian, Glass, Zoom and Motion; and Elastic transformation. Averaging over all corruptions we get an mCE gap of $1.9 \pm 0.9\%$ due to batch norm, or an increase of accuracy from $71.0 \pm 0.6\%$ to $72.9 \pm 0.7\%$.

VGG13 results were mostly consistent with VGG8: batch norm increased the error rate for all noise variants, at every intensity level. Particularly notable, the generalization gap enlarged to $26 - 28\%$ for Gaussian noise at severity levels 3, 4, and 5; and $17\%+$ for Impulse noise at levels 4 and 5. Averaging over all levels, we have gaps for noise variants of: Gaussian—$20.9 \pm 1.4\%$, Impulse—$13.6 \pm 0.6\%$, Shot—$14.1 \pm 1.0\%$, and Speckle—$11.1 \pm 0.8\%$. Robustness to the other corruptions seemed to benefit from the slightly higher clean test accuracy of $1.3 \pm 0.1\%$ due to batch norm for VGG13. The remaining generalization gaps varied from (negative) $0.2 \pm 1.3\%$ for Zoom blur, to $2.9 \pm 0.6\%$ for Pixelate. Overall mCE was reduced by $2.0 \pm 0.3\%$ for the unnormalized network.

For a WideResNet 28–10 (WRN) using "Fixup" initialization (Zhang et al., 2019b) to reduce the use of batch norm, the mCE was similarly reduced by $1.6 \pm 0.4\%$. Unpacking each category, the mean generalization gaps for noise were: Gaussian—$12.1 \pm 2.8\%$, Impulse—$10.7 \pm 2.9\%$, Shot—$8.7 \pm 2.6\%$, and Speckle—$6.9 \pm 2.6\%$. Note that the large uncertainty for these measurements is due to high variance for the model with batch norm, on average 2.3% versus 0.7% for Fixup. JPEG compression was next at $4.6 \pm 0.3\%$.

Interestingly, some corruptions that led to a positive gap for VGG8 showed a negative gap for the WRN, i.e. batch norm improved accuracy to: Contrast—$4.9 \pm 1.1\%$, Snow—$2.8 \pm 0.4\%$, Spatter—$2.3 \pm 0.8\%$. These were the same corruptions for which VGG13 lost, or did not improve its robustness when batch norm was removed, hence why we believe these correlate with standard test accuracy (highest for WRN). Visually, these corruptions appear to preserve texture information. Conversely, noise is applied in a spatially global way that disproportionately degrades these textures, emphasizing shapes and edges. It is now well known that modern CNNs trained on standard datasets have a propensity to rely excessively on texture rather than shape cues (Geirhos et al., 2019; Brendel & Bethge, 2019). The WRN obtains ≈ 0 training error and is in our view over-fitted; CIFAR-10 is known to be difficult to learn robustly given few samples (Schmidt et al., 2018).

D. Adversarial Spheres

The Adversarial Spheres dataset involves classification of concentric circles on the basis of their radius. This simple problem poses a challenge to the conventional wisdom regarding batch norm: not only does it harm robustness, it makes training less stable. We find, using the same architecture as in Gilmer et al. (2018), that training is significantly less stable with batch norm than without it, as the batch-normalized network is highly sensitive to the learning rate. We use SGD instead of Adam to avoid introducing unnecessary complexity, and especially since SGD has been shown to converge to the maximum-margin solution for linearly separable data (Soudry et al., 2018). We use a finite dataset of 500 samples from $\mathcal{N}(0, I)$ projected onto the circles.

To evaluate robustness on the 2D problem, we sample 10,000 test points from the same distribution for each class (20k total), and apply noise drawn from $\mathcal{N}(0, 0.005 \times I)$. We evaluate only the models that could be trained to 100% training accuracy with the smaller learning rate of $\eta = 0.001$. The model with batch norm classified 94.83% of these points correctly, while the unnormalized net obtained 96.06%.

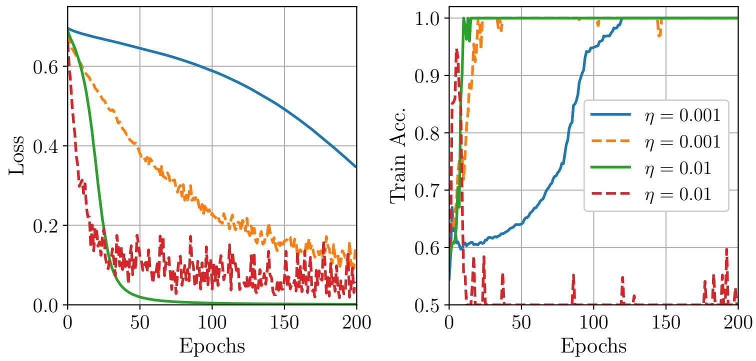


Figure 6. We train the same two-hidden-layer fully connected network of width 1000 units using the same mini-batch size of 50 as in Gilmer et al. (2018). We again turn batch norm on and off, on a 2D variant of the Adversarial Spheres binary classification problem. The model with batch norm fails to train for learning rate $\eta = 0.01$, which otherwise converges quickly for the unnormalized equivalent. Dashed lines denote the model with batch norm.

E. Qualitative Effect of Batch Normalization

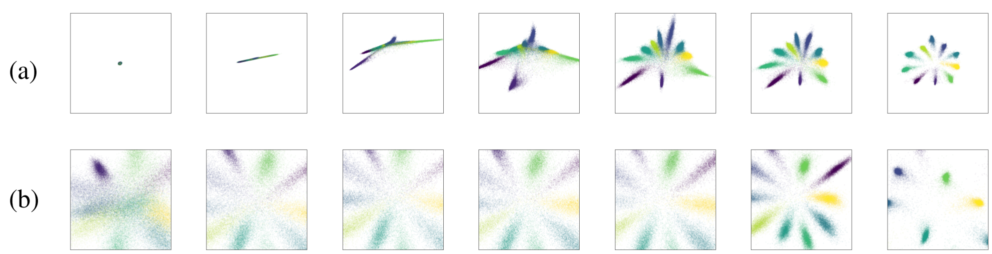


Figure 7. Visualization of activations in a two-neuron ($n = 2$) layer over 500 epochs. Architecture is a fully-connected MLP [(a) (784–392–196–2–49–10) and (b) (784–392–BN–196–BN–2–49–10)] with ReLU units, batch size 128, constant learning rate 1e-2, and weight decay λ=1e-3. The plots have a fixed x- and y-axis range of ± 10. All samples from the MNIST training set are plotted and colour coded by label.

We show a qualitative aspect of batch norm by visualizing clusters at the penultimate hidden layer of a fully connected network. Figure 7(a) depicts how the flow of information can be obstructed at initialization in relatively shallow unnormalized networks. Initially, there is no label-homogeneous partition and all data points are overlapping. The data points are spread further apart over the first ≈ 20 epochs (middle plot), with little emphasis on compression. Once it is possible to partition the clusters, a phase transition begins in which the clusters become tighter. Figure 7(b) introduces two batch-norm layers with default settings before the 2D layer. There are notable differences compared to Figure 7(a): i) the clusters are more easily partitioned at initialization as all data points are more spread out, thus facilitating faster training; ii) the clusters are more stationary and there is no clear phase transition; iii) the inter-cluster distance and the clusters themselves are larger, and the decision boundary more sensitive to small input variations.

F. Author Contributions

In the spirit of Sculley et al. (2018), we provide a summary of each author's contributions.

- First author formulated the hypothesis, conducted the experiments, and wrote the initial draft.
- Second author prepared detailed technical notes on the main references, met frequently with the first author to advance the work, and critically revised the manuscript.
- Third author originally conceived the key theoretical concept of Appendix B as well as some of the figures, and provided important technical suggestions and feedback.
- Fourth author met with the first author to discuss the work, and helped revise and improve the presentation of the manuscript.
- Senior author critically revised several iterations of the manuscript, helped improve the presentation, recommended additional experiments, and sought outside feedback.