Batch Normalization is a Cause of Adversarial Vulnerability

Anonymous authors
Paper under double-blind review

Abstract

Batch normalization (BN) 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 achieve low training error. However, it also reduces robustness to small adversarial input perturbations and common corruptions by double-digit percentages, as we show on five standard datasets. Furthermore, we find that substituting weight decay for BN is sufficient to nullify a relationship between adversarial vulnerability and the input dimension. A recent mean-field analysis found that BN induces gradient explosion when used on multiple layers, but this cannot fully explain the vulnerability we observe, given that it occurs already for a single BN layer. We argue that the actual cause is the tilting of the decision boundary with respect to the nearest-centroid classifier along input dimensions of low variance. As a result, the constant introduced for numerical stability in the BN step acts as an important hyperparameter that can be tuned to recover some robustness at the cost of standard test accuracy. We explain this mechanism explicitly on a linear "toy model" and show in experiments that it still holds for nonlinear "real-world" models.

1 Introduction

BN 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 during deployment to plausible, previously unseen inputs. 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 a challenge (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 BN—and patch 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; Madry 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 BN, 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 of various regularizers.

The frequently made observation that adversarial vulnerability can scale with the input dimension (Goodfellow et al., 2015; Gilmer et al., 2018; Simon-Gabriel et al., 2019) highlights the importance of identifying regularizers as more than merely a way to improve test accuracy. In particular, BN was a confounding factor in Simon-Gabriel et al. (2019), making the results of their initialization-time analysis hold after training. By adding ℓ_2 regularization and removing BN, we show that there is no *inherent* relationship between adversarial vulnerability and the input dimension.

2 Batch Normalization

We briefly review how BN modifies the hidden layers' pre-activations h of a neural network. We use the notation of Yang et al. (2019), where α is an index for units in a layer l, and i for a mini-batch of

B samples from the dataset; N_l denotes the number of units in layer l, W^l is the matrix of weights and b^l is 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 $\sigma_\alpha^2 = \frac{1}{B}\sum_i (h_{\alpha i} - \mu_\alpha)^2$. In the BN procedure, the mean μ_α is subtracted from the pre-activation of each unit $h_{\alpha i}^l$ (consistent with Ioffe & Szegedy (2015)), the result is divided by the standard deviation σ_α plus a small constant c to prevent division by zero, then scaled and shifted by the learned parameters γ_α and β_α, respectively. This is described in equation 1, where a per-unit nonlinearity ϕ, e.g., ReLU, is applied after the normalization.

$$h_i^l = W^l \phi(\tilde{h}_i^{l-1}) + b^l, \qquad \tilde{h}_{\alpha i}^l = \gamma_\alpha \frac{h_{\alpha i} - \mu_\alpha}{\sqrt{\sigma_\alpha^2 + c}} + \beta_\alpha. \tag{1}$$

This procedure introduces complications, however. Consider two mini-batches that differ by only a *single* example: due to the induced batch-wise nonlinearity, they will have different representations of *all* examples. These differences are amplified by stacking BN layers, and were shown to cause exploding gradients at initialization (Yang et al., 2019). Conversely, normalization of intermediate representations for two different training inputs impairs the ability to distinguish definite examples that ought to be classified with a large prediction margin (as judged by an "oracle"), from more ambiguous instances. The last layer of a discriminative neural network, in particular, is typically a linear decoding of class label-homogeneous clusters, and thus makes use of information contained in the mean and variance at this stage for classification. In light of these observations, we begin in our analysis by adding a single BN layer to models trained by gradient descent (GD). This is the most favorable scenario according to the analysis of Yang et al. (2019), where more layers and a smaller mini-batch size exacerbate the exploding gradients.

3 Boundary Tilting

Tanay & Griffin (2016) relate the adversarial vulnerability of linear classifiers to the tilting angle θ of the decision boundary w.r.t. the nearest-centroid classifier. Following their setup, we examine how BN affects this angle in a simple linear model, and then show that increasing model complexity cannot "undo" this vulnerability.

Consider the binary classification task of identifying two different types of input x subject to Gaussian noise with a linear classifier $w^\top x + b$. This can be modeled by the class-conditional distribution $p(x|y = j) = \mathcal{N}(\nu^j, \Sigma)$ with label $y \sim \mathrm{Ber}(0.5)$. The Bayes-optimal solution to this problem is given by the weight vector $w = \Sigma^{-1}(\nu^0 - \nu^1)$, and $b = \frac{1}{2}(\nu^1 + \nu^0)^\top \Sigma^{-1}(\nu^1 - \nu^0) + \log\frac{p(y=0)}{p(y=1)}$, where $p(y)$ denotes the marginal probability for the label y (see e.g. (Jordan, 1995)), while the nearest-centroid classifier is defined by $w^* = \nu^0 - \nu^1$.

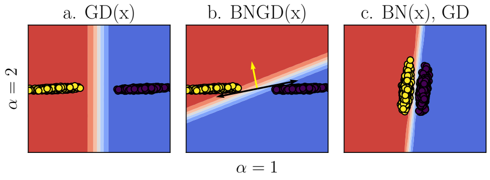


Figure 1: A dataset with one task-relevant ($\alpha = 1$) and one task-irrelevant dimension ($\alpha = 2$). Normalization aligns the decision boundary with the Bayes solution (indicated by arrows in "BNGD"), but this minimizes the averaged distance between the points and the boundary, maximizing adversarial vulnerability. Compared with the decision boundary of a linear model ($\theta \approx 0°$), the batch-normalized model has $\theta = 66.7°$. On the right is the dataset seen by the BNGD classifier. We use $\Sigma_{11} = 1$, $\Sigma_{22} = 0.01$, $\Sigma_{12} = \Sigma_{21} = 0.05$, $\nu^0 = [-5{,}0]$, and $\nu^1 = [5{,}0]$.

We analyze the effect of batch-normalizing the input to the classifier for this problem (i.e., $h_{\alpha i} = x_{\alpha i}$), first in the simplest setting where $\gamma_\alpha = 1, \beta_\alpha = 0 \; \forall \alpha$. We select the class distribution means ν^j to be symmetric around zero, so that the batch mean computed by BN is $\mu_\alpha = 0 \; \forall \alpha$. The batch-normalized linear classifier is thus defined as: $f(x) = \frac{w^\top x + b}{\sqrt{\sigma^2 + c}}$. By construction of our synthetic dataset, the variance of the batch can be deduced from the data parameters: $\sigma_\alpha^2 = (\nu_\alpha^j)^2 + \Sigma_{\alpha\alpha}$. The tilting angle θ of the batch-normalized decision boundary w.r.t. the one given by w^* (note that the boundary is perpendicular to w) is therefore approximately equal to the angle between the datasets before and after normalization. To compute θ, we divide the weights w by $\sqrt{\sigma^2 + c}$, and then normalize $w / \|w\|_2$, such that $\theta = \cos^{-1}(w^\top w^*)$. From this analysis it follows that the order of magnitude of c is important relative to the data variance: if

Table 1: As predicted by the theory, batch-normalized gradient descent (BNGD) yields a tilted decision boundary w.r.t. the nearest-centroid classifier, regardless of the affine parameters being learned or fixed. We report the tilting angle (θ) and accuracies of linear models trained on MNIST 3 vs. 7 for vanilla GD, GD with L2 weight decay "WD"($\lambda = 0.1$), and BNGD. Affine = "F" indicates $\gamma = 1$ and $\beta = 0$, whereas "T" means they are randomly initialized and learnable. AWGN $= \mathcal{N}(0,1)$, FGSM used with $\epsilon = 1/10$. Entries are the mean and its standard error over five random seeds.

Model	Test Acc.	AWGN Acc.	FGSM Acc.	$\theta \in [0,90°]$
GD	96.94 ± 0.08	90.08 ± 0.07	66.96 ± 0.49	49.04 ± 0.46
GD + WD	96.93 ± 0.05	91.93 ± 0.14	74.20 ± 0.35	40.83 ± 0.46
BNGD Affine F	97.75 ± 0.03	49.67 ± 0.18	0.15 ± 0.02	90.00 ± 0.00
BNGD Affine T	97.40 ± 0.07	49.50 ± 0.20	0.13 ± 0.02	90.00 ± 0.00

$c > \sigma_\alpha^2$ then the effective weight value w_α is reduced, and if $c < \sigma_\alpha^2$ and σ_α^2 is small, then w_α increases greatly, causing boundary tilting along direction α.

We depict simulations of the toy model in Figure 1. We use constant learning rate GD, which is known to converge to the max-margin solution—equivalent to the nearest centroid classifier in this case–for linear models on separable data (Soudry et al., 2018). Batch-normalized GD (BNGD) converges for arbitrary learning rates for linear models (Cai et al., 2019); we use a value of 0.1 for 1000 epochs.

Next, we train linear models on the MNIST 3 vs. 7 dataset with 5000 training samples (drawn uniformly per class) using a learning rate of 0.1 for 50 epochs. We compute the angle θ w.r.t. the nearest-centroid classifier, which is obtained by subtracting the "average 3" from the "average 7" of the full training set. Although this may seem like a crude reference point, the nearest-centroid classifier is much more robust than the linear model of Goodfellow et al. (2015), achieving 40% accuracy for the fast gradient sign method (FGSM) at $\epsilon = 1/4$ vs. $\approx 0\%$. Results consistent with the boundary tilting theory are shown in Table 1, which not only shows that BN causes tilting, but that this is unaffected by the parameters γ and β. Post-normalization, there is no signal to γ and β about the variances of the original dataset. This is consistent with other works that observe γ and β do not influence the studied effect (van Laarhoven, 2017; Zhang et al., 2019a; Yang et al., 2019)

Increasing the numerical stability constant c increases robustness in terms of absolute test accuracy for additive white Gaussian noise (AWGN) on MNIST and CIFAR-10 datasets by 33% and 41% respectively (at the cost of standard accuracy). For brevity, we defer the experimental details and full results to Appendix A.

4 Empirical Results

For the main practical results, we evaluate the robustness (quantified as the drop in test accuracy under input perturbations) of convolutional networks, with and without BN.[1] Although the use of BN affects the range of suitable training hyperparameters, standard procedures from the literature were used that were originally tuned for batch-normalized models. The datasets – MNIST, SVHN, CIFAR-10, and ImageNet – were normalized to zero mean and unit variance.

As a white-box adversarial attack we use projected gradient descent (PGD), ℓ_∞- and ℓ_2-norm variants, for its simplicity and ability to degrade performance with little perceptible change to the input (Madry et al., 2018). We run PGD for 20–40 iterations, with $\epsilon_\infty = 0.03$ and a step size of $\epsilon_\infty/10$ for SVHN, CIFAR-10, and $\epsilon_\infty = 0.01$ for ImageNet. For PGD-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", as well as the CIFAR-10-C common corruption benchmark (Hendrycks & Dietterich, 2019). We found these methods were sufficient to demonstrate a considerable disparity

[1] Unless stated otherwise, we leave the internal parameters of BN to their default values as in modern deep learning frameworks, c=1e-5 with γ and β enabled.

in robustness due to BN, but this is not intended as a formal security evaluation. All uncertainties are the standard error of the mean.[2]

Table 2: Test accuracies of VGG8 on SVHN.

BN	Clean	Noise	PGD-ℓ_∞	PGD-ℓ_2
✗	92.60 ± 0.04	83.6 ± 0.2	27.1 ± 0.3	22.0 ± 0.8
✓	94.46 ± 0.02	78.1 ± 0.6	10 ± 1	1.6 ± 0.3

For SVHN, 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 2, finding that BN increased clean test accuracy by $1.86\pm0.05\%$, and reduced test accuracy for additive noise by $5.5\pm0.6\%$, for PGD-ℓ_∞ by $17\pm1\%$, and for PGD-ℓ_2 by $20\pm1\%$.

Table 3: Test accuracies of VGG8 and WideResNet–28–10 on CIFAR-10 and CIFAR-10.1 (`v6`) in several variants: clean, noisy, and PGD perturbed.

		CIFAR-10				CIFAR-10.1	
Model	BN	Clean	Noise	PGD-ℓ_∞	PGD-ℓ_2	Clean	Noise
VGG	✗	87.9 ± 0.1	79 ± 1	52.9 ± 0.6	65.6 ± 0.3	75.3 ± 0.2	66 ± 1
VGG	✓	88.7 ± 0.1	73 ± 1	35.7 ± 0.3	59.7 ± 0.3	77.3 ± 0.2	60 ± 2
WRN	F	94.6 ± 0.1	69 ± 1	20.3 ± 0.3	9.4 ± 0.2	87.5 ± 0.3	68 ± 1
WRN	✓	95.9 ± 0.1	58 ± 2	14.9 ± 0.6	8.3 ± 0.3	89.6 ± 0.2	58 ± 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. We evaluate two families of contemporary models: one without skip connections (VGG) and a WideResNets (WRN) using "Fixup" initialization (Zhang et al., 2019b) to reduce the use of BN.

In the first experiment, a basic comparison with and without BN shown in Table 3, we evaluate the best model in terms of test accuracy after training for 150 epochs with a fixed learning rate of 0.01. In this case, inclusion of BN for VGG 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 PGD perturbations by $17.3\pm0.7\%$ and $5.9\pm0.4\%$ for ℓ_∞ and ℓ_2 variants, respectively. Very similar results are obtained on a new test set, CIFAR-10.1 `v6` (Recht et al., 2018): BN slightly improves the clean test accuracy (by $2.0\pm0.3\%$), but leads to a considerable drop in test accuracy of $6\pm1\%$ for the case with additive noise, and $15\pm1\%$ and $3.4\pm0.6\%$ respectively for ℓ_∞ and ℓ_2 PGD variants (PGD absolute values omitted for CIFAR-10.1 in Table 3 for brevity).

Table 4: VGG models of increasing depth on CIFAR-10, with and without BN (BN). See text for differences in hyperparameters compared to Table 3.

Model		Test Accuracy (%)		
L	BN	Clean	Noise	PGD-ℓ_∞
8	✗	89.29 ± 0.09	81.7 ± 0.3	55.6 ± 0.4
8	✓	90.49 ± 0.01	77 ± 1	40.6 ± 0.6
13	✗	91.74 ± 0.02	77.8 ± 0.7	40.3 ± 0.7
13	✓	93.0 ± 0.1	67 ± 1	28.5 ± 0.4
16	✓	92.8 ± 0.1	66 ± 2	28.9 ± 0.2
19	✓	92.65 ± 0.09	68 ± 2	30.0 ± 0.1

It has been suggested that one of the benefits of BN 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 is increased to 0.1 when BN is used. We prolong training for up to 350 epochs, and drop the learning rate by a factor of ten at epoch 150 and 250 in both cases, which increases clean test accuracy relative to results in Table 3. The deepest model that is trainable using standard "He" initialization (He et al., 2015) without BN is VGG13. [3] None of the deeper batch-normalized models re-

[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 first three successful runs for consistency with the other experiments.

Table 5: Robustness of three modern convolutional neural network architectures with and without BN on the `CIFAR-10-C` common "noise" corruptions (Hendrycks & Dietterich, 2019). We use "F" to denote the Fixup variant of WRN. Values were averaged over five intensity levels for each corruption.

Model		Test Accuracy (%)				
Variant	BN	Clean	Gaussian	Impulse	Shot	Speckle
VGG8	✗	$87.9{\pm}0.1$	$\mathbf{65.6{\pm}1.2}$	$\mathbf{58.8{\pm}0.8}$	$\mathbf{71.0{\pm}1.2}$	$\mathbf{70.8{\pm}1.2}$
	✓	$88.7{\pm}0.1$	$56.4{\pm}1.5$	$51.2{\pm}0.1$	$65.4{\pm}1.1$	$66.3{\pm}1.1$
VGG13	✗	$91.74{\pm}0.02$	$\mathbf{64.5{\pm}0.8}$	$\mathbf{63.3{\pm}0.3}$	$\mathbf{70.9{\pm}0.4}$	$\mathbf{71.5{\pm}0.5}$
	✓	$93.0{\pm}0.1$	$43.6{\pm}1.2$	$49.7{\pm}0.5$	$56.8{\pm}0.9$	$60.4{\pm}0.7$
WRN28	F	$94.6{\pm}0.1$	$\mathbf{63.3{\pm}0.9}$	$\mathbf{66.7{\pm}0.9}$	$\mathbf{71.7{\pm}0.7}$	$\mathbf{73.5{\pm}0.6}$
	✓	$95.9{\pm}0.1$	$51.2{\pm}2.7$	$56.0{\pm}2.7$	$63.0{\pm}2.5$	$66.6{\pm}2.5$

cover the robustness of the most shallow, or same-depth unnormalized equivalents, nor does the higher learning rate with BN 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 D.

We also evaluate robustness on the common corruption benchmark comprising 19 types of real-world effects that can be grouped into four categories: "noise", "blur", "weather", and "digital" corruptions (Hendrycks & Dietterich, 2019). Each corruption has five "severity" or intensity levels. We report the mean error on the corrupted test set (mCE) by averaging over all intensity levels and corruptions (Hendrycks & Dietterich, 2019). We summarize the results for two VGG variants and a WideResNet on CIFAR-10-C, trained from scratch on the default training set for three and five random seeds, respectively. Accuracy for the noise corruptions, which caused the largest difference in accuracy with BN, are outlined in Table 5.

The key takeaway is: *For all models tested, the batch-normalized variant has a higher error rate for all corruptions of the "noise" category, at every intensity level.*

Averaging over all 19 corruptions we find that BN increases mCE by $1.9 \pm 0.9\%$ for VGG8, $2.0{\pm}0.3\%$ for VGG13, and $1.6{\pm}0.4\%$ for WRN. There is a large disparity in accuracy when modulating BN for different corruption categories, therefore we examine these in more detail in Appendix F.

Interestingly, some corruptions that led to a positive gap for VGG8 show a negative gap for the WRN, i.e., BN improved accuracy to: Contrast—$4.9 \pm 1.1\%$, Snow—$2.8 \pm 0.4\%$, Spatter—$2.3 \pm 0.8\%$. These are the same corruptions for which VGG13 loses, or does not improve its robustness when BN is removed. We suspect accuracy for these corruptions correlates with standard test accuracy, which is highest for the 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 known that modern CNNs trained on standard image datasets have a propensity to rely heavily on texture in addition to shape and edge cues for object recognition (Geirhos et al., 2019). We evaluate pre-trained bag-of-local-feature models (BagNets) on

Table 6: Robustness of pre-trained ImageNet models with and without BN. *Note*: The numeric suffix indicates number of layers, or the spatial patch width in pixels (of 224) for BagNet.

		Top 5 Test Accuracy (%)		
Model	BN	Clean	Noise	PGD-ℓ_∞
VGG11	✗	88.63	49.16	37.12
VGG11	✓	89.81	49.95	26.12
VGG19	✗	90.88	64.86	34.19
VGG19	✓	91.84	68.79	24.49
AlexNet	✗	79.07	41.41	39.12
ResNet18	✓	88.65	79.62	31.07
BagNet-9	✓	70.39	1.25	7.42
BagNet-17	✓	81.16	5.09	16.66
BagNet-33	✓	86.99	14.62	24.34

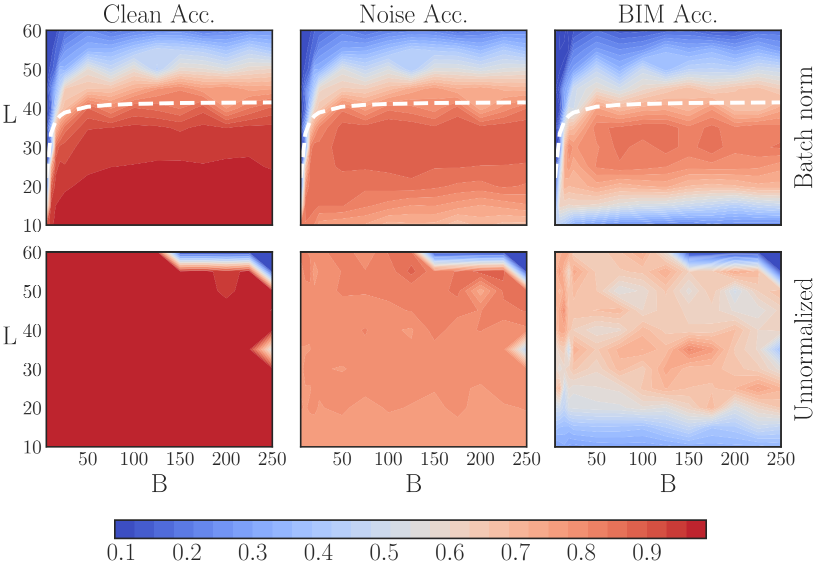


Figure 2: We extend the experiment of Yang et al. (2019) by training fully-connected nets of depth L and constant-width ($N_l = 384$) ReLU layers by SGD, batch size B, and learning rate $\eta = 10^{-5}B$ on MNIST. The BN parameters γ and β were left as default, momentum disabled, and $c = 10^{-3}$. The dashed line is the theoretical maximum trainable depth of batch-normalized networks as a function of the batch size. We report the clean test accuracy, and that for additive Gaussian noise and BIM perturbations. The batch-normalized models were trained for 10 epochs, while the unnormalized ones were trained for 40 epochs as they took longer to converge. The 40 epoch batch-normalized plot was qualitatively similar with dark blue bands for BIM for shallow and deep variants. The dark blue patch for 55 and 60 layer unnormalized models at large batch sizes depicts a total failure to train. These networks were trainable by reducing η, but for consistency we keep η the same in both cases.

ImageNet with an architecture that discards spatial information between patches and is thus considered to make extensive use of texture patterns for classification (Brendel & Bethge, 2019). For patch sizes $\{9,17,33\}$, the top-5 accuracies of the BagNets are reduced to just 1.25%, 5.09%, and 14.62% for AWGN, respectively. Compared with Table 6, where all models obtain over 40%, these figures suggest that robustness to Gaussian noise is a good proxy for the use of texture for ImageNet classification. Our results support the hypothesis that BN may be exacerbating this tendency to leverage superficial texture-like information for classification of image data.

Next, we evaluate the robustness of pre-trained ImageNet models from the `torchvision.models` repository, which conveniently provides models with and without BN.[4] Results are shown in Table 6, where BN improves top-5 accuracy on noise in some cases, but consistently reduces it by 8.54% to 11.00% (absolute) for PGD. The trends are the same for top-1 accuracy, only the absolute values are smaller; the degradation varies from 2.38% to 4.17%. Given the discrepancy between noise and PGD for ImageNet, we include a black-box transfer analysis in the Appendix D.2 that is consistent with the white-box analysis.

Finally, we explore the role of batch size and depth in Figure 2. We find that BN 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.

[4] `https://pytorch.org/docs/stable/torchvision/models.html`, `v1.1.0`.

5 Vulnerability and Input Dimension

A recent work Simon-Gabriel et al. (2019) analyzes adversarial vulnerability of batch-normalized networks at initialization time and conjectures based on a scaling analysis that, under the commonly used He et al. (2015) initialization scheme, adversarial vulnerability scales as $\sim \sqrt{d}$. 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) (Madry et al., 2018), with a modest trade-off of clean accuracy.

Table 7: Evaluating the robustness of a MLP with and without batch norm. See text for architecture. 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

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 loss scaling as the norm of the input increases with d. We increase the MNIST image width $\sqrt{d}$ from 28 to 56, 84, and 112 pixels. The loss $\mathcal{L}$ is predicted to grow like $\sqrt{d}$ for ϵ-sized attacks by Thm. 4 of Simon-Gabriel et al. (2019). We confirm that without regularization the loss does scale roughly as predicted: the predicted values lie between loss ratios obtained for $\epsilon = 0.05$ and $\epsilon = 0.1$ attacks for most image widths (see Table 4 of Appendix E). 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 more detailed explanation and results are provided in Appendix E.

Next, we repeat this experiment with a two-hidden-layer ReLU MLP, with the number of hidden units equal to the half the input dimension, and optionally use one hidden layer with batch norm.[5] 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.[6]

Table 8: Evaluating the robustness of a MLP with ℓ_2 weight decay (same λ as for linear model, see Table 5 of Appendix E). See text for architecture. 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

Despite a difference in clean accuracy of only $0.08\pm0.05\%$, Table 7 shows that for the original image resolution, batch norm reduced accuracy for noise by $16.4\pm0.4\%$, and for BIM-ℓ_∞ by $43.8\pm0.5\%$. 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.

We then apply the ℓ_2 regularization constants tuned for the respective input dimensions on the linear model to the ReLU MLP with no further adjustments. Table 8 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 induce a test error increase of $1.69\pm0.01\%$, which is substantial on MNIST.

[5] This choice of architecture is mostly arbitrary, the trends were the same for constant width layers.

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

Furthermore, using the same regularization constant and no batch norm increases clean test accuracy by $1.39 \pm 0.04\%$, and for the BIM-ℓ_∞ perturbation by $21.7 \pm 0.4\%$.

Finally, following the guidance in the original work on batch norm (Ioffe & Szegedy, 2015) to the extreme ($\lambda = 0$): when we *reduce* weight decay when using batch norm, accuracy for the $\epsilon_\infty = 0.1$ perturbation is degraded by $79.3 \pm 0.3\%$ for $\sqrt{d} = 56$, and $81.2 \pm 0.2\%$ for $\sqrt{d} = 84$.

In all cases, using batch norm greatly reduced test accuracy for noisy and adversarially perturbed inputs, while weight decay increased accuracy for such inputs.

6 Related Work

Our work examines the effect of batch norm on model robustness at test time. References with an immediate connection to our work were discussed in the previous sections; here we briefly mention other works that do not have a direct relationship to our experiments, but are relevant to batch norm in general.

The original work Ioffe & Szegedy (2015) that introduced batch norm as a technique for improving neural network training and test performance motivated it by the "internal covariate shift" – a term referring to the changing distribution of layer outputs, an effect that requires subsequent layers to steadily adapt to the new distribution and thus slows down the training process. Several follow-up works started from the empirical observation that batch norm usually accelerates and stabilizes training, and attempted to clarify the mechanism behind this effect. One argument is that batch-normalized networks have a smoother optimization landscape due to smaller gradients immediately before the batch-normalized layer (Santurkar et al., 2018). However, Yang et al. (2019) study the effect of stacking many batch-normalized layers and prove that this causes gradient explosion that is exponential in network depth for networks without skip connections and holds for any non-linearity. In practice, relatively shallow batch-normalized networks seem to benefit from the "helpful smoothing" of the loss surface property Santurkar et al. (2018), while very deep networks are not trainable (Yang et al., 2019). In our work, we found that a single batch-normalized layer suffices to induce severe adversarial vulnerability.

Weight decay's loss scaling mechanism is complementary to other mechanisms identified in the literature, for instance that it 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 (between training and test error), even in batch-normalized networks where it is presumed to have no effect. Given that batch norm is not typically used on all layers, the loss scaling mechanism persists, although to a lesser degree in this case.

Shafahi et al. (2019) performed similar input dimension scaling experiments as in this work and came to a similar conclusion that the input dimension is irrelevant to adversarial vulnerability. However, like Simon-Gabriel et al. (2019), they use PGD rather than weight decay to prevent vulnerability from increasing with input dimension. Although it can be shown that robust optimization is equivalent to parameter norm regularization for linear models if we allow the ϵ-ball (aka disturbance δ) to vary with each example (Xu et al., 2009), we maintain that the latter is a more efficient approach.

7 Conclusion

We found that there is no free lunch with batch norm when model robustness is a concern: the accelerated training properties and occasionally higher clean test accuracy come at the cost of increased vulnerability, 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.

References

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

Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding Batch Normalization. In *Advances in Neural Information Processing Systems 31*, pp. 7705–7716. Curran Associates, Inc., 2018.

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

Yongqiang Cai, Qianxiao Li, and Zuowei Shen. A Quantitative Analysis of the Effect of Batch Normalization on Gradient Descent. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 882–890. PMLR, 2019.

Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: From adversarial to random noise. In *Advances in Neural Information Processing Systems 29*, pp. 1632–1640. Curran Associates, Inc., 2016.

Angus Galloway, Thomas Tanay, and Graham W. Taylor. Adversarial Training Versus Weight Decay. *arXiv preprint arXiv:1804.03308*, 2018.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019.

Justin Gilmer, Luke Metz, Fartash Faghri, Sam Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial Spheres. In *International Conference on Learning Representations Workshop Track*, 2018.

Ian. J. Goodfellow, Jonathon. Shlens, and Christian. Szegedy. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*, 2015.

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

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

Elad Hoffer, Itay Hubara, and Daniel Soudry. 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. Curran Associates, Inc., 2017.

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

Jöern-Henrik Jacobsen, Jens Behrmann, Nicholas Carlini, Florian Tramèr, and Nicolas Papernot. Exploiting Excessive Invariance caused by Norm-Bounded Adversarial Robustness. *Safe Machine Learning workshop at ICLR*, 2019.

Michael I. Jordan. Why the logistic function? A tutorial discussion on probabilities and neural networks. *MIT Computational Cognitive Science Report*, 1995.

Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial Machine Learning at Scale. *International Conference on Learning Representations*, 2017.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*, 2018.

Norman Mu and Justin Gilmer. MNIST-C: A robustness benchmark for computer vision. In *International Conference on Machine Learning Workshop on Uncertainty and Robustness in Deep Learning*, volume abs/1906.02337, 2019.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical Black-Box Attacks Against Machine Learning. In *Asia Conference on Computer and Communications Security*, ASIA CCS, pp. 506–519, Abu Dhabi, UAE, 2017. ACM.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do CIFAR-10 Classifiers Generalize to CIFAR-10? *arXiv:1806.00451*, 2018.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How Does Batch Normalization Help Optimization? In *Advances in Neural Information Processing Systems 31*, pp. 2488–2498. 2018.

Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the first adversarially robust neural network model on MNIST. In *International Conference on Learning Representations*, 2019.

Ali Shafahi, W. Ronny Huang, Christoph Studer, Soheil Feizi, and Tom Goldstein. Are adversarial examples inevitable? In *International Conference on Learning Representations*, 2019.

Carl-Johann Simon-Gabriel, Yann Ollivier, Leon Bottou, Bernhard Schölkopf, and David Lopez-Paz. First-Order Adversarial Vulnerability of Neural Networks and Input Dimension. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5809–5817. PMLR, 2019.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, and Nathan Srebro. The Implicit Bias of Gradient Descent on Separable Data. In *International Conference on Learning Representations*, 2018.

Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. 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. Springer International Publishing, 2018. ISBN 978-3-030-01258-8.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

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

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness May Be at Odds with Accuracy. In *International Conference on Learning Representations*, 2019.

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

Huan Xu, Constantine Caramanis, and Shie Mannor. Robustness and Regularization of Support Vector Machines. *Journal of Machine Learning Research*, 10:1485–1510, 2009. ISSN 1532-4435.

Greg Yang, Jeffrey Pennington, Vinay Rao, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. A Mean Field Theory of Batch Normalization. In *International Conference on Learning Representations*, 2019.

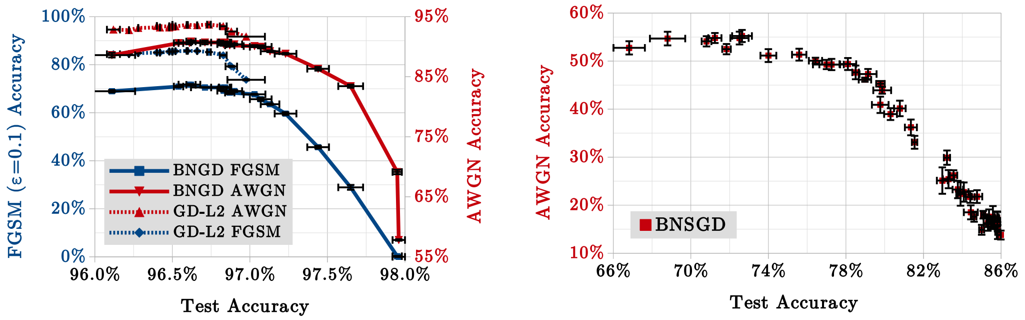
Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger Grosse. Three Mechanisms of Weight Decay Regularization. In *International Conference on Learning Representations*, 2019a.

Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. Residual Learning Without Normalization via Better Initialization. In *International Conference on Learning Representations*, 2019b.

A The Numerical Stability Constant

The constant c originally added to the mini-batch variance in the denominator for numerical stability (named ϵ in Ioffe & Szegedy (2015)) turns out to be an important hyperparameter in terms of robustness. It acts as a threshold on the variance of all input dimensions or neurons. When c is much less than the minimum variance over dimensions, it induces boundary tilting along the low-variance dimensions. In Figure 3 we sweep c for MNIST 3 vs. 7 and CIFAR-10, and compare the corresponding clean test accuracy with FGSM and AWGN accuracy for MNIST, and AWGN for CIFAR-10. For MNIST, increasing c allows us to trade-off clean accuracy for robustness to FGSM, but is suboptimal compared to L2 weight decay. For these experiments we fixed $\gamma_\alpha = 1$ and $\beta_\alpha = 0$.

For CIFAR-10, eight-layer VGG models were trained with a constant learning rate of 0.01 with no drops, momentum of 0.9, a batch size of 128, and 50 epochs (for computational reasons) over four random seeds. As for BNGD, for this particular experiment we apply BN only to the input layer. A consistent trend is observed where robustness to noise increases greatly as c is increased, but we note that this occurs for c several orders of magnitude greater than default settings.



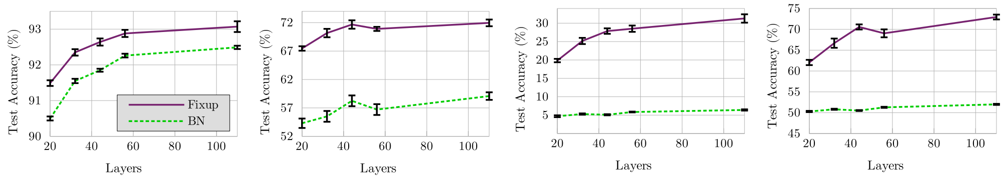
(a)

(b)

Figure 3: Sweeping the BN numerical stability constant for (a) the MNIST 3 vs. 7 dataset with $c \in$ [1e-3, 2e+1] for BNGD, and as a baseline the L2 regularization constant $\lambda \in$ [1e-3, 9] (shown as "GD-L2"). (b) Sweeping c for batch-normalized SGD (BNSGD) on the CIFAR-10 dataset with $c \in$ [1e-6, 3e+3] for the VGG8 architecture. Increasing either c or λ has a similar effect to trade clean test accuracy for increased robustness, until the effect is too large and both accuracies degrade. The absolute accuracies are consistently higher without BN. Error bars indicate standard error of the mean over four and five random seeds for MNIST and CIFAR-10, respectively. We recommend following each curve from right to left to be consistent with our description above. The default setting (highest clean test accuracy, lowest robustness) starts in the bottom right corner and the initial trade-off between clean test accuracy and robustness is traced up and leftwards until the curves inflect.

B On an Accuracy vs. Robustness Trade-Off

It is natural to wonder if the degradation in robustness arising from the use of BN is simply due to BN increasing the standard test accuracy, given a known trade-off between the two (Tanay & Griffin, 2016; Galloway et al., 2018; Su et al., 2018; Tsipras et al., 2019). Note that if the relationship between input X and label Y is free of noise, e.g., as in Gilmer et al. (2018), then there is no such trade-off and increasing accuracy corresponds to increasing robustness. For the toy problem we studied in § 3,



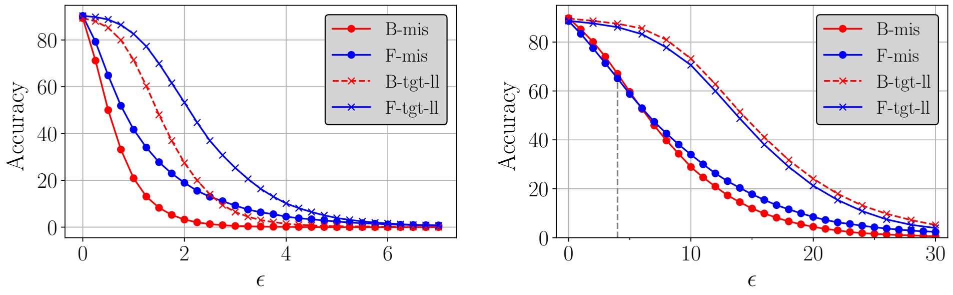
(a)

(b)

(c)

(d)

Figure 4: Accuracy of batch-normalized versus unnormalized (Fixup) residual networks of varying depth. Models were trained with standard hyperparameters and evaluated on CIFAR-10: (a) clean test set, (b) noisy test set, (c) PGD ℓ_∞, and (d) PGD ℓ_2. Error bars denote the standard error of the mean over five random seeds. The unnormalized networks obtain higher accuracy for all tests and depths.



(a) (b)

Figure 5: Test accuracy vs. ϵ for (a), a naturally trained ResNet32 and (b)a state-of-the-art WideResNet 28-10 CIFAR-10 baseline (Madry et al., 2018) trained with PGD-ℓ_∞ ($\epsilon_{\max} = 4$, 5 iterations, step size 1 out of 255, w/rand. start) in batch-normalized ('B') and unnormalized ('F' for Fixup) variants. At test-time, 20 PGD iterations are used (the "strong" adversary from Madry et al.) for the ℓ_∞, and ℓ_2 norms. A misclassification objective ('mis') is compared to targeting the least-likely label ('tgt-ll').

BN actually aligned the decision boundary with the Bayes-optimal solution, so increasing standard accuracy may be intrinsic to the normalization itself in some cases.

Given that BN does typically increase clean test accuracy by some small amount on commonly used datasets, we thought it was most representative to not intentionally limit the performance of BN. We did, however, find natural cases where BN did *not* improve clean test accuracy. We trained ResNets{20,32,44,56,110} using Fixup initialization on CIFAR-10: all consistently obtain about 0.5% higher clean test accuracy than their batch-normalized equivalent, and are also more robust to noise ($\approx 15\%$) and PGD ℓ_∞ and ℓ_2 perturbations ($\approx 30\%$), as shown in Figure 4.

For MNIST, the results of Tables 6 & 7 also show compatible clean accuracy irrespective of BN, and yet vastly different robustness. Thus, the vulnerability induced by BN is not merely a consequence of increasing standard test accuracy.

C Adversarial Training and Accuracy vs. ϵ-Curves

For brevity, we opted to report accuracy for an arbitrary small value of ϵ in the main text. In general, however, it is more useful to plot accuracy vs. ϵ to ensure the accuracy reaches zero for reasonably large ϵ to help rule out gradient masking issues (Papernot et al., 2017; Athalye et al., 2018). This also shows that ϵ was not cherry-picked.

Figure 5(b) shows that PGD-ℓ_∞ training recovers much of the BN-vulnerability gap when tested on PGD-ℓ_∞, but there is a still a non-trivial improvement at $\epsilon = {}^{8}/_{255}$ from 38.84% to 41.57% (recall that we only trained with $\epsilon_{\max} = {}^{4}/_{255}$, so absolute accuracy is slightly lower than in Madry et al. (2018)). Ultimately, adversarial robustness is concerned with robustness to the worst attack in our threat model. If we consider the `contrast` corruption from (Hendrycks & Dietterich, 2019), PGD training reduces accuracy by 23.5% and 28.5% for Fixup and BN respectively. We do not believe it is reasonable for a threat model of natural image classifiers to exclude natural changes in image contrast. Increasing capacity combined with adversarial training therefore does not solve the robustness issue, and can exacerbate other vulnerabilities in the model (Jacobsen et al., 2019; Mu & Gilmer, 2019).

Similar results are observed on MNIST with 40 iterations of PGD training, step size of 0.01, and $\epsilon_{\max} = {}^{0.3}/_{1.0}$. Here, PGD training reduces overall accuracy on MNIST-C (Mu & Gilmer, 2019) by 4.28%, and an additional 3.82% when BN is used.

D Additional Empirical Results

This section contains supplementary explanations and results to those of Section 4.

D.1 Additional SVHN and CIFAR-10 Results for Deeper Models

Our first attempt to train VGG models on SVHN with more than eight layers failed, therefore for a fair comparison we report the robustness of the deeper models that were only trainable by using BN in Table 9. None of these models obtained much better robustness in terms of PGD-ℓ_2, although they did better for PGD-ℓ_∞.

Table 9: VGG variants on SVHN with BN.

	Test Accuracy (%)			
L	Clean	Noise	PGD-ℓ_∞	PGD-ℓ_2
11	95.31 ± 0.03	80.5 ± 1	20.2 ± 0.2	6.1 ± 0.2
13	95.88 ± 0.05	77.2 ± 7	21.7 ± 0.5	5.4 ± 0.2
16	94.59 ± 0.05	78.1 ± 4	19.2 ± 0.3	3.0 ± 0.2
19	95.1 ± 0.3	78 ± 1	24.2 ± 0.6	4.1 ± 0.4

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 BN. Note that the Fixup variant still contains one BN layer before the classification layer, but the number of BN layers is still greatly reduced.[7]

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

	CIFAR-10				CIFAR-10.1	
Model	Clean	Noise	PGD-ℓ_∞	PGD-ℓ_2	Clean	Noise
Fixup	94.6 ± 0.1	69.1 ± 1.1	20.3 ± 0.3	9.4 ± 0.2	87.5 ± 0.3	67.8 ± 0.9
BN	95.9 ± 0.1	57.6 ± 1.5	14.9 ± 0.6	8.3 ± 0.3	89.6 ± 0.2	58.3 ± 1.2

We train WideResNets (WRN) with five unique seeds and report their test accuracies in Table 10. 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 comparable with 67.8 ± 0.9, and 57.6 ± 1.5 with 58.3 ± 1.2. The Fixup variant improves accuracy by $11.6 \pm 1.9\%$ for noisy CIFAR-10, $9.5 \pm 1.5\%$ for noisy CIFAR-10.1, $5.4 \pm 0.6\%$ for PGD-ℓ_∞, and $1.1 \pm 0.4\%$ for PGD-ℓ_2.

We believe our work serves as a compelling motivation for Fixup and other techniques that aim to reduce usage of BN. The role of skip-connections should be isolated in future work since absolute values were consistently lower for residual networks.

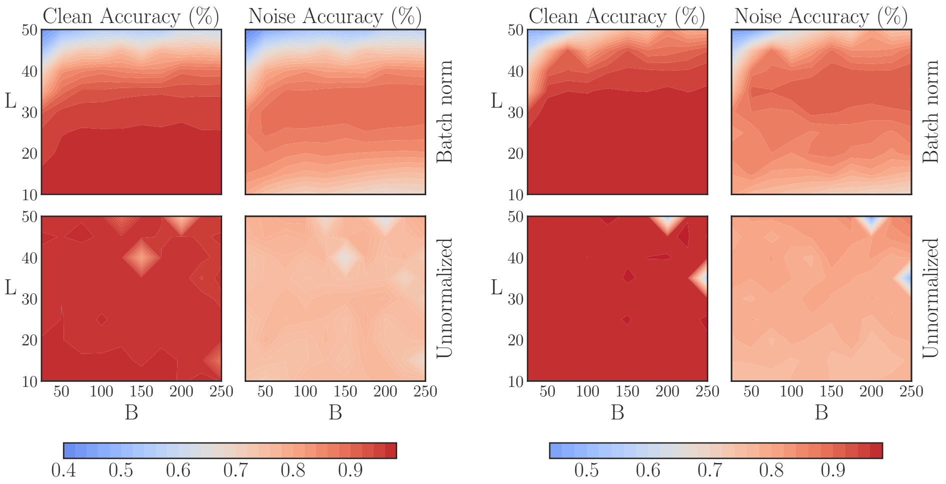
D.2 ImageNet Black-box Transferability Analysis

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 11 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.

[7] 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.

Table 11: ImageNet validation accuracy for adversarial examples transfered between VGG variants of various depths, indicated by number, with and without BN ("✓", "✗"). 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



(a)

(b)

Figure 6: 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 BN 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.

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.

D.3 Batch Norm Limits Maximum Trainable Depth and Robustness

In Figure 6 we show that BN 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 6(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 6(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 epochs total. When they do converge, we see more configurations that achieve higher clean test accuracy than batch-normalized networks in Figure 6(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 mostly non-overlapping in Figure 6(b).

E Weight Decay and Input Dimension

Consider a logistic classification model represented by a neural network consisting of a single unit, parameterized by weights $w \in \mathbb{R}^d$ and bias $b \in \mathbb{R}$, with input denoted by $x \in \mathbb{R}^d$ and true labels $y \in \{\pm 1\}$. Predictions are defined by $s = w^\top x + b$, and the model is optimized through empirical risk minimization, i.e., by applying stochastic gradient descent (SGD) to the loss function equation 2, where $\zeta(z) = \log(1+e^{-z})$:

$$\mathbb{E}_{x,y\sim p_{\text{data}}}\,\zeta(y(w^\top x + b)). \tag{2}$$

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 equation 2 is equivalent to minimizing

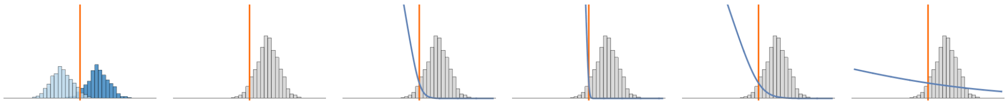
$$\mathbb{E}_{x,y\sim p_{\text{data}}}\,\zeta(\|w\|_2 \times y d(x)). \tag{3}$$

We define the *scaled loss* as

$$\zeta_{\|w\|_2}(z) := \zeta(\|w\|_2 \times z) \tag{4}$$

and note that adding a ℓ_2 regularization term in equation 3, resulting in equation 5, can be understood as a way of controlling the scaling of the loss function:

$$\mathbb{E}_{x,y\sim p_{\text{data}}}\,\zeta_{\|w\|_2}(y d(x)) + \lambda \|w\|_2 \tag{5}$$



(a) $w^\top x$ (b) $y(w^\top x)$ (c) $\zeta(y(w^\top x))$ (d) $\zeta_5(yd(x))$ (e) $\zeta_{0.5}(yd(x))$ (f) $\zeta_{0.05}(yd(x))$

Figure 7: (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 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.

In Figures 7(a)-7(c), we develop intuition for the different quantities contained in equation 2 with respect to a typical binary classification problem, while Figures 7(d)-7(f) depict the effect of the regularization parameter λ on the scaling of the loss function.

To test this theory empirically we study a model with a single linear layer (number of units equals input dimension) and cross-entropy loss function on variants of MNIST of increasing input dimension, to approximate the toy model described in the "core idea" from Simon-Gabriel et al. (2019) as closely as possible, but with a model capable of learning. 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 mini-batch size

Table 12: 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. (2019), we use ϵ-FGSM perturbations, the optimal ℓ_∞ attack for a linear model. Values in rows with $\sqrt{d} > 28$ are ratios of entry (accuracy or loss) wrt the $\sqrt{d} = 28$ baseline. "Pred." is the predicted increase of the loss $\mathcal{L}$ due to a small ϵ-perturbation using Thm. 4 of Simon-Gabriel et al. (2019).

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

of 128. In Table 12 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", defined as the average increase of the loss after attack, which is predicted to grow like $\sqrt{d}$ by Theorem 4 of Simon-Gabriel et al. (2019), 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. (2019) note that independence between vulnerability and the 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 parameter λ chosen dependent on d to correct for the loss scaling. This way we recover input dimension invariant vulnerability with little degradation of test accuracy, e.g., see the result for $\sqrt{d} = 112$ and $\epsilon = 0.1$ in Table 12: the accuracy ratio is 1.00 ± 0.03 with weight decay regularization, 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.[8] 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, albeit to a lesser degree.

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

Table 13: Two-hidden-layer ReLU MLP (see main text for architecture), with and without batch norm (BN), trained for 50 epochs and repeated over five random seeds. Values in rows with $\sqrt{d}>28$ are ratios wrt the $\sqrt{d}=28$ baseline (accuracy or loss). There is a considerable increase 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 $61\pm1\%$ degradation in absolute accuracy compared to the baseline.

Model		(Relative) Test Accuracy		(Relative) Loss	
$\sqrt{d}$	BN	Clean	$\epsilon=0.1$	Clean	$\epsilon=0.1$
28	✗	$97.95\pm0.08\%$	$66.7\pm0.9\%$	0.0669 ± 0.0008	1.06 ± 0.02
28	✓	0.9992 ± 0.0012	0.34 ± 0.03	1.06 ± 0.04	3.18 ± 0.03
56	✗	1.0025 ± 0.0009	0.80 ± 0.02	0.87 ± 0.02	1.68 ± 0.03
56	✓	1.0027 ± 0.0008	0.13 ± 0.09	0.91 ± 0.03	5.83 ± 0.03
84	✗	1.0033 ± 0.0009	0.71 ± 0.02	0.86 ± 0.02	2.15 ± 0.03
84	✓	1.0033 ± 0.0010	0.09 ± 0.08	0.88 ± 0.02	7.34 ± 0.02

F Common Corruption Robustness

For VGG8, the mean generalization gaps due to batch norm for noise were: Gaussian—$9.2\pm1.9\%$, Impulse—$7.5\pm0.8\%$, Shot—$5.6\pm1.6\%$, and Speckle—$4.5\pm1.6\%$. After the “noise” category the next most damaging corruptions (by difference in accuracy due to batch norm) were: Contrast—$4.4\pm1.3\%$, Spatter—$2.4\pm0.7\%$, JPEG—$2.0\pm0.4\%$, and Pixelate—$1.3\pm0.5\%$. 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.

For VGG13, the batch norm accuracy gap enlarged to $26-28\%$ for Gaussian noise at severity levels 3, 4, and 5; and over 17% for Impulse noise at levels 4 and 5. Averaging over all levels, we have gaps for noise variants of: Gaussian—$20.9\pm1.4\%$, Impulse—$13.6\pm0.6\%$, Shot—$14.1\pm1.0\%$, and Speckle—$11.1\pm0.8\%$. Robustness to the other corruptions seemed to benefit from the slightly higher clean test accuracy of $1.3\pm0.1\%$ for the batch-normalized VGG13. The remaining generalization gaps varied from (negative) $0.2\pm1.3\%$ for Zoom blur, to $2.9\pm0.6\%$ for Pixelate.

For the WRN, the mean generalization gaps for noise were: Gaussian—$12.1\pm2.8\%$, Impulse—$10.7\pm2.9\%$, Shot—$8.7\pm2.6\%$, and Speckle—$6.9\pm2.6\%$. 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 $4.6\pm0.3\%$.

G Adversarial Spheres

The “Adversarial Spheres” dataset contains points sampled uniformly from the surfaces of two concentric n-dimensional spheres with radii $R=1$ and $R=1.3$ respectively, and the classification task is to attribute a given point to the inner or outer sphere. We consider the case $n=2$, that is, datapoints from two concentric circles. This simple problem poses a challenge to the conventional wisdom regarding batch norm: not only does batch norm harm robustness, it makes training less stable. In Figure 9 we show that, using the same architecture as in Gilmer et al. (2018), the batch-normalized network is highly sensitive to the learning rate η. 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. The unormalized network achieves zero training error for η up to 0.1 (not shown), whereas the batch-normalized network is already untrainable at $\eta=0.01$. To evaluate robustness, 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 classifies 94.83% of these points correctly, while the unnormalized net obtains 96.06%.

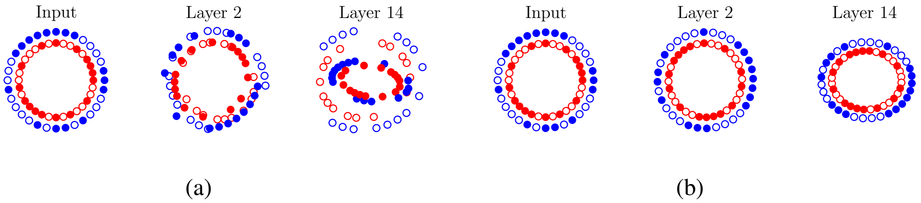


Figure 8: Two mini-batches from the “Adversarial Spheres” dataset (2D variant), and their representations in a deep linear network at initialization time (a) with batch norm and (b) without batch norm. Mini-batch membership is indicated by marker fill and class membership by colour. Each layer is projected 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.

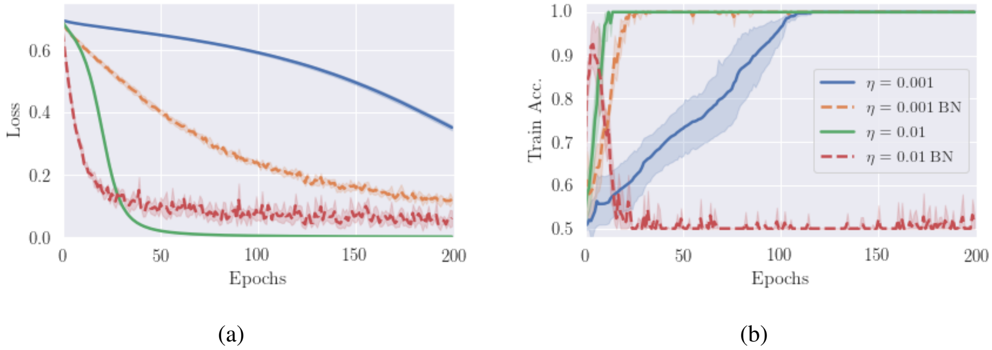


Figure 9: We train the same two-hidden-layer fully connected network of width 1000 units using ReLU activations and a mini-batch size of 50 on a 2D variant of the “Adversarial Spheres” binary classification problem Gilmer et al. (2018). Dashed lines denote the model with batch norm. The batch-normalized model fails to train for a learning rate of $\eta = 0.01$, which otherwise converges quickly for the unnormalized equivalent. We repeat the experiment over five random seeds, shaded regions indicate a 95% confidence interval.