Received 21 December 2022, accepted 26 January 2023, date of publication 1 March 2023, date of current version 11 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3250661

RESEARCH ARTICLE

Adversarial Attacks and Batch Normalization: A Batch Statistics Perspective

AWAIS MUHAMMAD[1], FAHAD SHAMSHAD[2], AND SUNG-HO BAE[1], (Member, IEEE)
[1]Department of Computer Science and Engineering, Kyung-Hee University, Yongin 17104, South Korea
[2]Computer Vision Department, Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, United Arab Emirates

Corresponding author: Sung-Ho Bae (shbae@khu.ac.kr)

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korean Government [Ministry of Science and ICT (MSIT)] (Artificial Intelligence Innovation Hub) under Grant 2021-0-02068; in part by the IITP Grant funded by the Korean Government (MSIT) under Grant 2022-0-00759; and in part by the IITP Grant funded by the Korean Government (MSIT), Artificial Intelligence Convergence Innovation Human Resources Development (Kyung Hee University), under Grant RS-2022-00155911.

ABSTRACT Batch Normalization (BatchNorm) is an effective architectural component in deep learning models that helps to improve model performance and speed up training. However, it has also been found to increase the vulnerability of models to adversarial attacks. In this study, we investigate the mechanism behind this vulnerability and took first steps towards a solution called RobustNorm. We observed that adversarial inputs tend to shift the distributions of the output of the BatchNorm layer, leading to inaccurate train-time statistics and increased vulnerability. Through a series of experiments on various architectures and datasets, we confirm our hypothesis. We also demonstrate the effectiveness of RobustNorm in improving the robustness of models under adversarial perturbation while maintaining the benefits of BatchNorm.

INDEX TERMS Batch normalization, adversarial robustness, transfer learning.

I. INTRODUCTION

Deep learning [46] models have shown impressive performance on a wide variety of tasks computer vision, such as image recognition [31], [42], [76], object detection [27], [39], [58], [59], semantic segmentation [10], [52], etc. However, these deep models are vulnerable to small and imperceptible perturbations in the input [8], [28], [54], also known as adversarial examples. These small changes in the input can fool the state-of-the-art models and degrade their performance significantly. The adversarial vulnerability of the deep models has serious security and robustness consequences [6], [44], [50].

A plethora of research has been dedicated to understand the reasons for this intriguing behavior of neural networks [15], [25], [28], [34], [37], [64], [66]. These explanations include inherent linearity of non-linear deep models [28], excessive invariance [37], sensitivity to input distribution [15], non-robust yet highly predictive features in the input space [34], etc. However, Galloway et al. [23] studied this problem from an architectural angle. They empirically showed that BatchNorm also contributes to this vulnerability. {In this study, we investigate the mechanism behind this vulnerability and took first steps towards a solution called RobustNorm.

BatchNorm [36] was designed to reduce the internal covariate shift by normalizing the input of each layer with batch statistics (mean and variance). However, batch statistics are only available during training since their calculation needs batched inputs. To circumvent this, BatchNorm keeps an estimate of population statistics during training and utilizes these estimated statistics during inference. In standard training, population statistics are estimated from clean inputs. We {hypothesized that the inherent distribution shift present in adversarial inputs makes these estimated statistics inaccurate as they are estimated for clean images.

{To validate our hypothesis, we conducted a series of experiments using various architectures, diverse attack methods, and various levels of perturbation. Specifically, we replaced the train-time estimated batch statistics with validation-set statistics calculated using adversarial inputs. Our experiments on a range of datasets and architectures demonstrated a significant improvement in robustness. These

The associate editor coordinating the review of this manuscript and approving it for publication was Prakasam Periasamy.

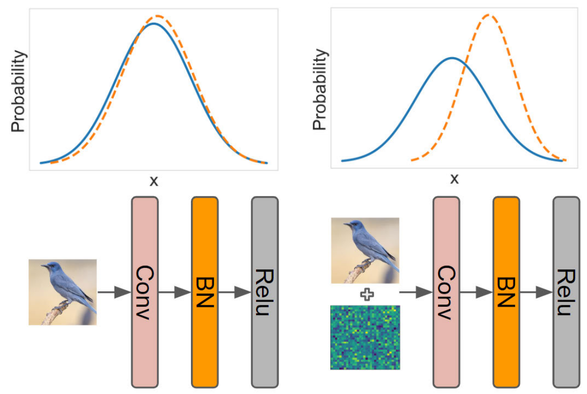


FIGURE 1. A conceptual illustration of the effect of adversarial distribution shift on BatchNorm. In the plot, blue line represents an ideal distribution that BatchNorm expects and orange line shows the data distribution of inference-time examples. Input distribution is a good approximate ideal distribution for clean images, but the distribution gets shifted when adversarial noise is added to the input image. This shift invalidates the implicit assumption of BatchNorm that the train and validation data will be from the same distribution. This makes population statistics estimated during training (with clean images) inaccurate and contributes to the adversarial vulnerability of neural networks.

results support our observation that the use of incorrect batch statistics can affect the vulnerability of models to adversarial attacks. We further explored the generalizability of this trend under various conditions, including different architectures, perturbation levels, and adversarial training, to confirm the validity of our hypothesis.

{Based on our observations, we propose that a normalization approach that does not rely on the estimation of population statistics may improve the robustness of models. To address this, we propose an improved variant of BatchNorm called RobustNorm. Our experiments demonstrate that RobustNorm performs well in terms of robustness while retaining the other benefits of BatchNorm.

We also extended our hypothesis for transfer learning. In transfer learning, models trained on the source dataset (pre-trained models) are often used as feature extractors. A classifier is then trained on these features extracted for the target dataset. The feature extractor uses batch statistics that are estimated from the source dataset. We show that by updating batch statistics on the target dataset, we can achieve significant accuracy gains. We also observed that the accuracy increase depends on the similarity between the source and target dataset. Concisely, our contributions are as follows.

- {We investigated how BatchNorm causes adversarial vulnerability in deep models by conducting formulating the shift hypothesis and conducting a diverse set of experiments to validate this effect
- {Based on our observations, we took first towards a more robust normalization approach
- We showed that our batch-norm-explanation can be extended beyond adversarial attacks by illustrating results on transfer learning

II. RELATED WORK

Since the inception of adversarial attacks for neural networks [66], many explanations have been proposed to understand the adversarial vulnerability of neural networks. Szegedy et al. [66] linked adversarial vulnerability to blind spots in the discontinuous classification boundary of the neural network, Goodfellow et al. [28] showed that it is because of the local linearity of neural networks. Recent works have connected adversarial vulnerability with many factors like random noise [19], [22], spurious correlations learned by neural networks [34], insufficient data [61], high dimensions of input data [20], [26], distributional shift [15], [37] etc. Our work is different in the sense that we do not try to understand the broader phenomenon but rather the contribution of a BatchNorm. Our work is related to Galloway et al. [23] who empirically showed that BatchNorm is one cause of the adversarial vulnerability of neural networks. However, our focus is on understanding how BatchNorm causes this.

{Recent works have explored the effect of BatchNorm and estimated statistics on the adversarial vulnerability of deep models. To start with, Benz et al. [5] investigated the contribution of Non-Robust Features (NRFs) in increasing the performance of models. They show that BatchNorm's use of NRFs is the predominant reason for the improvement in the performance of deep models. The second line of work has also utilized a hypothesis similar to ours for different purposes. Xie and Yuille [72] proposed a two-domain hypothesis for clean and adversarial images and showed that it could be leveraged to improve adversarial training. Xie et al. [75] showed that by using different batch statistics for clean and adversarial images during training, it is possible to achieve state-of-the-art ImageNet results without any extra data. Schneider et al. [62] illustrated that a model can achieve better robustness against many common corruptions by replacing statistics estimated from clean images with the statistics estimated from corrupted images.

BatchNorm [36] was introduced to reduce the internal covariant shift of deep models. It improved the stability and optimization of neural networks. Since then, many different variants of BatchNorm have also been proposed. Each BatchNorm variant intends to solve a particular problem of the original formulation. LayerNorm [4] solves the problem of fixed batch size training making it useful in sequence models; BatchReNorm [35] and GroupNorm [70] eases the problem of small-batch training making it functional for tasks like detection or segmentation, and InstanceNorm [67] reduces intra-batch dependency making it applicable in style transfer.

Many other variants such as Normalization Propagation [2], Batch Re-normalization [35], instance normalization [67], batch instance normalization [67], Kalman batch normalization [69], decorrelated normalization [32], switch normalization [48], sparse switchable normalization [63], switchable whitening [56], dynamic normalization [49] has also been introduced recently to overcome different issues. Similarly, efforts have been made to extend BatchNorm to other domains as well [7], [12], [24], [53], [60], [70]. However, our work is different from these works as we took first a more robust normalization approach.

III. PRELIMINARIES

We consider a supervised classification task for data $\{\boldsymbol{x}, y\}_{i=1}^{n}$. Our goal is to learn a feature extractor $\mathbf{f} = F(\mathbf{x})$ and a classifier C such that $y = C(\mathbf{f}, \boldsymbol{w})$. In adversarial settings, the objective of the adversary is to add small additive perturbation $\boldsymbol{\delta} \in \mathbb{R}^n$ in clean image $\boldsymbol{x}$: $\boldsymbol{x}_{adv} = \boldsymbol{x} + \boldsymbol{\delta}$ while satisfying following constraints. First, adversarial image should follow a perturbation budget ϵ: $\|\boldsymbol{x}_{adv} - \boldsymbol{x}\|_p \leq \epsilon$. Second, the adversarial image ($\boldsymbol{x}_{adv}$) should look visually similar to the original image $\boldsymbol{x}$. Third, the trained model should incorrectly label i.e., $C(F(\boldsymbol{x}_{adv})) \neq y$.

A. ROBUSTNESS EVALUATION

Adversarial accuracy (or adversarial robustness) is accuracy of model on adversarially perturbed test set. Generally, we evaluate robustness of a model with PGD-20 attack with $\alpha = 2/255$ and reported ϵ value following standard practice [51]. However, adversarial examples can be generated in many different ways. To make our results more rigorous, we also used diverse set of adversarial attack methods. We have used different variants of gradient based attacks: Fast Gradient Sign Method (FGSM) [28], Basic Iterative Method (BIM) [44], Projected Gradient Descent (PGD) [51], Momentum Iterative fast gradient sign Method (MIM) [17], Carlini-Wagner attack (CW) [8]. {For adversarial attack budget (ϵ), we use a scale of 0-255 for color images (CIFAR and ImageNet) and 0-1 for black and white images (MNIST). The different scale is based on range of value different datasets use.

For PGD attack, we also report results for two versions: one with 20 iterations and one with 100 iterations. We also use a parameter-free and reliable attack called Auto-PGD [13] or APGD-CE. To show that our results are not effected by gradient masking, we also used query-based blackbox attack called Square attack [1]. The square attack do not use model information (e.g., gradient) and, therefore, it is immune to problems like gradient masking. We set number of queries to 5000 for all of our evaluations.

B. PURPOSE OF THE WORK

Following [9], here we describe the purpose of our work. The intention of this work is neither to propose a new defense mechanism like [51] nor a broader explanation for the adversarial phenomenon like [34] etc. Instead, our purpose is to understand the contribution of a specific component of CNNs in this vulnerability. For this reason, we have used ϵ values smaller than commonly used for some of our experiments. However, we also report results on borad range of ϵ.

C. TRANSFER LEARNING

For transfer learning, we considered that the feature extractor $\mathbf{f} = F(x)$ is already trained on the source dataset (ImageNet) and we want to learn a classifier C for the target dataset on top of it.

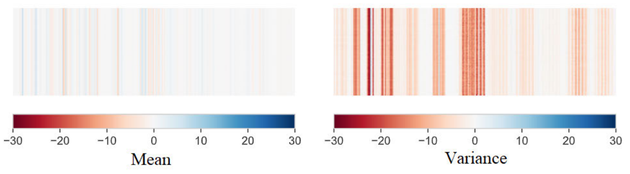


FIGURE 2. The difference between validation-set batch statistics estimated population statistics under adversarial attack: $\mu_B - \hat{\mu}_{\mathcal{P}}$, $\sigma_B^2 - \hat{\sigma}_{\mathcal{P}}$. The x-axis represents channels of the network and y-axis represents batches. Each line shows difference of estimated and calculated value for one channel.

IV. HOW DOES BatchNorm CAUSE ADVERSARIAL VULNERABILITY

BatchNorm estimates the population statistics during training by using moving averages of batch statistics. These estimated values are used during inference. However, one inherent assumption of this process is that training and inference data come from the same underlying distribution. Adversarial attacks introduce a targeted shift in the distribution of input data. This shift breaks this assumption. In the following sections, we explain how BatchNorm works, and empirically demonstrate our hypothesis through various experiments.

TABLE 1. The effect of using batch statistics (B) vs estimated population statistics (P) on adversarial accuracy. Batch statistics are calculated from one batch of images (batch to be classified) from the validation set and estimated population statistics are estimated during training. The results are shown for five different datasets and five different attacks. We have used ResNet18 for ImageNet and ResNet20 for all other datasets. This table proves our hypothesis by showing an increase in accuracy if we use batch statistics that more representative of an adversarial shift in distribution.

Stat.	Normal	FGSM	BIM	MIM	CW	PGD
MNIST ($\epsilon = 0.2/1$, $c = 10$)						
P	99.2	59.6	7.7	16.7	6.9	07.7
B	99.1	**91.7**	**69.3**	**78.7**	**46.2**	**69.3**
Fashion-MNIST ($\epsilon = 0.06/1$, $c = 10$)						
P	93.7	41.7	1.5	2.8	02.2	1.5
B	93.6	**73.8**	**32.4**	**41.5**	**25.8**	**32.3**
CIFAR10 ($\epsilon = 2/255$, $c = 0.01$)						
P	92.1	48.3	23.1	27.0	23.7	23.1
B	87.2	**67.3**	**46.7**	**54.3**	**40.7**	**46.6**
CIFAR100 ($\epsilon = 2/255$, $c = 0.01$)						
P	68.9	20.2	07.3	08.5	08.8	07.3
B	58.7	**31.3**	**18.6**	**22.1**	**29.0**	**18.5**
ImageNet ($\epsilon = 1/255$, $c = 0.01$)						
P	62.7	12.5	4.0	5.2	3.4	4.4
B	**63.5**	**29.2**	**18.0**	**21.4**	**21.0**	**19.0**

A. HOW BatchNorm WORKS

In this section, we briefly explain the working principle of BatchNorm. BatchNorm uses batch statistics during training and estimated population statistics during inference.

TABLE 2. The effect of using population vs batch statistics on adversarial accuracy for several models on CIFAR10 dataset. The column Stat. shows type of statistics used for BatchNorm, P stands for Population Statistics ($\hat{\mu}_{\mathcal{P}}, \hat{\sigma}^2_{\mathcal{P}}$) and B stands for Batch Statistics calculated from validation batch at inference ($\mu_{\mathcal{B}}$ and $\sigma^2_{\mathcal{B}}$).

Attack	ϵ	ResNet20		ResNet38		ResNet50		VGG11		VGG16		WRN16-10	
		P	B	P	B	P	B	P	B	P	B	B	P
None	0	92.01	89.62	92.58	90.50	92.82	87.64	91.83	90.11	93.06	91.79	95.13	93.65
PGD-20	1	43.8	**47.71**	47.28	50.39	53.26	48.86	70.81	69.72	64.59	61.53	55.38	61.58
	2	7.67	18.15	9.73	21.38	14.07	22.93	42.70	49.45	27.14	27.79	14.34	31.57
	4	0.45	2.28	0.73	3.95	0.85	4.74	10.13	17.54	2.91	2.84	0.09	6.15
	8	0.03	0.13	0.07	0.54	0.08	0.67	2.52	1.49	0.2	0.2	0.02	0.51
PGD-100	1	43.3	46.6	46.65	49.41	52.84	48.17	70.77	69.71	64.39	61.47	55.03	61.22
	2	6.69	15.69	8.84	18.62	12.95	20.89	42.28	49.16	26.53	27.01	13.74	30.35
	4	0.39	1.55	0.54	2.57	0.79	3.57	9.32	16.78	2.54	2.48	0.71	5.36
	8	0.02	0.11	0.05	0.29	0.07	0.38	2.1	1.26	0.07	0.15	0.07	0.43
APGD-CE	1	12.73	15.42	15.01	18.88	20.72	27.18	60.16	58.52	48.04	41.52	24.22	30.3
	2	0.22	2.2	0.39	3.08	0.77	6.22	24.54	26.91	9.33	7.62	0.75	4.87
	4	0	1.68	0	1.5	0.0	2.59	2.02	3.22	0.07	1.64	0.0	1.4
	8	0	2.13	0	2.03	0.0	2.87	0.45	2.21	0.0	2.35	0.0	1.5

1) BATCH STATISTICS

Consider a mini-batch $\mathcal{B}$ of size M, containing samples $\boldsymbol{x}_i$ for $i = 1, 2, \ldots, M$. In the training procedure, the normalized feature maps $\hat{\boldsymbol{x}}_i$ are computed as:

$$\hat{\boldsymbol{x}}_i = \frac{\boldsymbol{x}_i - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}}}, \tag{1}$$

where batch statistics are the sample mean $\mu_{\mathcal{B}}$ and sample variance $\sigma^2_{\mathcal{B}}$ computer over the batch $\mathcal{B}$ as:

$$\mu_{\mathcal{B}} = \frac{1}{M}\sum_{i=1}^{M}\boldsymbol{x}_i; \quad \sigma^2_{\mathcal{B}} = \frac{1}{M}\sum_{i=1}^{M}(\boldsymbol{x}_i - \mu_{\mathcal{B}})^2. \tag{2}$$

Besides, a pair of values γ, β are used to shift and scale the normalized value $\hat{\boldsymbol{x}}_i$ as:

$$y_i = \gamma\hat{\boldsymbol{x}}_i + \beta. \tag{3}$$

For the sake of simplicity, we will omit this in our future discussions.

2) ESTIMATED POPULATION STATISTICS

During inference, it is not possible to calculate batch statistics ($\mu_{\mathcal{B}}$ and $\sigma_{\mathcal{B}}$) as only one sample is available. To circumvent this problem, BatchNorm needs an estimate of population statistics. This estimate of population statistics is computed by maintaining the moving averages of the batch statistics during training. Formally, the moving average (also called tracking) of mean and variance are computed as follows:

$$\hat{\mu}_{\mathcal{P}} = (1-\tau)\hat{\mu}_{\mathcal{P}} + \tau\mu_{\mathcal{B}}, \quad \hat{\sigma}^2_{\mathcal{P}} = (1-\tau)\hat{\sigma}^2_{\mathcal{P}} + \tau\sigma^2_{\mathcal{B}}, \tag{4}$$

where $\hat{\mu}_{\mathcal{P}}$ are estimated population means, $\hat{\mu}_{\mathcal{P}}$ are estimated values of population variance, and τ is a hyper-parameter that weighs previous moving average and current batch statistics. These population statistics are used during inference as:

$$\hat{\boldsymbol{x}}_{\text{test}} = \frac{\boldsymbol{x}_{\text{test}} - \hat{\mu}_{\mathcal{P}}}{\hat{\sigma}_{\mathcal{P}}}. \tag{5}$$

B. DEVIL IS IN THE ESTIMATED STATISTICS

During the inference, BatchNorm normalizes the input with $\hat{\mu}_{\mathcal{P}}$ and $\hat{\sigma}^2_{\mathcal{P}}$. These statistics are estimated on clean inputs during training. The adversarial attack introduces a shift in the input and makes the estimated $\hat{\mu}_{\mathcal{P}}$ and $\hat{\sigma}_{\mathcal{P}}$ inaccurate. We show a hypothetical depiction of this idea in Figure 1.

To show this difference, we forward propagated all the validation set samples of CIFAR10 with PGD adversarial attack and calculated batch statistics (μ_B, σ^2_B) of each channel of a trained ResNet20. Their difference with estimated population statistics ($\hat{\mu}_{\mathcal{P}}$ and $\hat{\sigma}^2_{\mathcal{P}}$) is shown in Figure 2 where x-axis represents channels and the y-axis represents the difference for validation batches. The figure shows that estimated population statistics do not match the batch statistics under the PGD attack.

Recent works [15], [38] have also highlighted the link between the shift in the input data distribution and robustness. The same observation has also been used to augment adversarial examples to get SOTA results for image recognition [73]. Based on these observations, we made the following hypothesis:

Hypothesis: BatchNorm's population statistics ($\hat{\mu}_{\mathcal{P}}$ and $\hat{\sigma}_{\mathcal{P}}$) are estimated from $(\mathbf{x}, y) \sim \mathbb{P}$ and an implicit assumption is that inference images will also come from the same distribution. However, the addition of adversarial noise $\boldsymbol{\delta}$ in clean images shifts this distribution. This shift makes population statistics inaccurate. The use of these incorrect statistics makes a neural network with BatchNorm more vulnerable to adversarial inputs.

To empirically validate this hypothesis, we conducted multiple experiments in different settings. We describe these experiments in the following sections.

1) EXPERIMENT 1 — REPLACING TRAIN-TIME ESTIMATED STATISTICS WITH VALIDATION-SET BATCH STATISTICS

One way to verify our hypothesis is to use an adversarially perturbed validation set to calculate the batch statistics instead of train-time estimated population statistics. This means replacing $\hat{\mu}_{\mathcal{P}}, \hat{\sigma}^2_{\mathcal{P}}$ with $\mu_{\mathcal{B}_v}, \sigma^2_{\mathcal{B}_v}$ where

$\mathcal{B}_v = \{x_1^v, \ldots, x_M^v\}$ is the mini-batch of perturbed validation set. Note that this is only possible for a large enough validation set. Our only purpose here is to show the validity of our hypothesis.

We report results for five different datasets and five different adversarial attacks in Table 1. For MNIST and Fashion-MNIST, we use an 8-layer ResNet. We trained these two models for 50 epochs with a learning rate of 0.1. The learning rate is decreased by 10× at the 30-th epoch. For CIFAR10 and CIFAR100 experiments, we use a ResNet20. We train it for 150 epochs. The default learning rate of 0.1 is decreased at 80-th and 120-th epochs. For ImageNet, we use a ResNet18 and trained it for 100 epochs. The default learning rate of 0.1 is decreased at 30-th, 60-th, and 90-th epochs. We follow Section III for robustness evaluation.

The clean accuracy decreases when we use BatchNorm with batch statistics, so we may expect a similar decrease in adversarial accuracy. However, we observe an increase. For instance, on MNIST, we get 7% BIM adversarial accuracy with population statistics but replacing them with batch statistics from the validation set increase this to 69%. A similar effect is also visible across all the attacks, datasets, and training modes.

2) EXPERIMENT 2 — EVALUATION ON DIVERSE ATTACKS, ARCHITECTURES AND PERTURBATION LEVELS

The previous experiment uses ResNet architectures. To make this evaluation more rigorous, we repeated the previous experiments (e.g., replacing estimated statistics with batch statistics) on diverse sets of architectures, more challenging adversarial attacks, and larger perturbation budgets.

For this experiment, we use three different types of architecture. First, we use standard ResNet [31] with three different depths: 20, 38, and 50. Second, we use WideResNet [77] with a depth of 16 and a width of 10. The WideResNet are similar to ResNet in general architecture, but they have a larger capacity as the number of channels is increased significantly. Third, we also use VGG [65] with depths of 11 and 16. VGG is significantly different from ResNet as it does not use skip connections.

For robustness evaluation, we use three attacks: PGD-20, PGD-100, and APGD-CE. PGD-20 and PGD-100 use 20 and 100 iterations of gradient descent to find adversarial attacks. APGD-CE attack is a more reliable attack and it does not require any parameters. All of these attacks are generated with four perturbation (ϵ) levels: 1, 2, 4, 8.

The results are shown in Table 2. The P column stands for population statistics, i.e., standard BatchNorm layer, and the B column stands for using batch statistics instead of population statistics. The robustness of models using batch statistics is higher at all perturbation levels. For instance, the PGD-20 robustness of a ResNet50 with $\epsilon = 2$ is 14.03 when the model uses population statistics. But, it increases to 22.93. The same model has APGD-CE robustness of 0.77 at the same perturbation level as population statistics. This robustness increases to 6.22% when the same model uses batch statistics. These results shows that our hypothesis holds across different architectures, adversarial attacks, and perturbation levels.

3) EXPERIMENT 3 — ROBUSTNESS OF VARIANTS OF BATCHNORM

Based on different intuitions and insights, many alternatives to BatchNorm have been introduced. Some of these variants do not require estimation of population statistics e.g., layer normalization [4], Fixup Initialization [33] etc. Our hypothesis suggests that the adversarial accuracy of these variants should be higher than BatchNorm. To test this, we performed experiments with three different variants of BatchNorm for CIFAR10. The results are shown in Table 3. The clean accuracy of these alternatives is less than BatchNorm, so we should expect a similar drop in adversarial accuracy. On the contrary, there is an increment of adversarial accuracy. For instance, if we train a neural network with LayerNorm, clean accuracy decreases from 92.1% to 89.4%. However, adversarial robustness for PGD attacks increases significantly from 23.1% to 30.1%. This increment in robustness shows the role of estimated statistics in the vulnerability of CNNs.

TABLE 3. Effects of replacing BatchNorm with alternatives that do not require any moving average, therefore immune to adversarial shift. Although all of these alternatives have inferior clean accuracy; they always outperform BatchNorm in adversarial accuracy.

Norm	Normal	FGSM	BIM	MIM	CW	PGD
No Norm	82.9	43.4	29.0	31.5	28.5	29.0
Fixup Init.	91.4	55.7	37.1	41.1	17.9	39.0
LayerNorm	89.4	50.8	30.1	33.4	31.1	30.1
BatchNorm	92.1	48.4	23.1	27.0	23.7	23.8

4) EXPERIMENT 4 — ADVERSARIAL TRAINING

Adversarial training leverages adversarially perturbed examples to train a neural network. An adversarially trained BatchNorm layer estimates population statistics with adversarial examples. Therefore, we should expect better adversarial accuracy, which has already been shown [51]. We should also expect a smaller gap between using population statistics and input batch statistics (the setup we have used in Experiment 1). This indeed is true and adversarial training bridges the gap between estimated population statistics and validation set batch statistics based BatchNorm, as shown in Table 4. For instance, on the CIFAR10 dataset, the gap between BatchNorm with validation set batch statistics and population statistics is 100% for regular training, but it shrinks to 30% for adversarial training. These results show the importance of the reliability of train-time estimated population statistics and their effect on the adversarial performance of a neural network.

5) EXPERIMENT 5 — GRADIENT MASKING AND BLACKBOX ATTACK

Recently, [3] showed that obfuscated gradients may give a false sense of better robustness. To show that our experiments

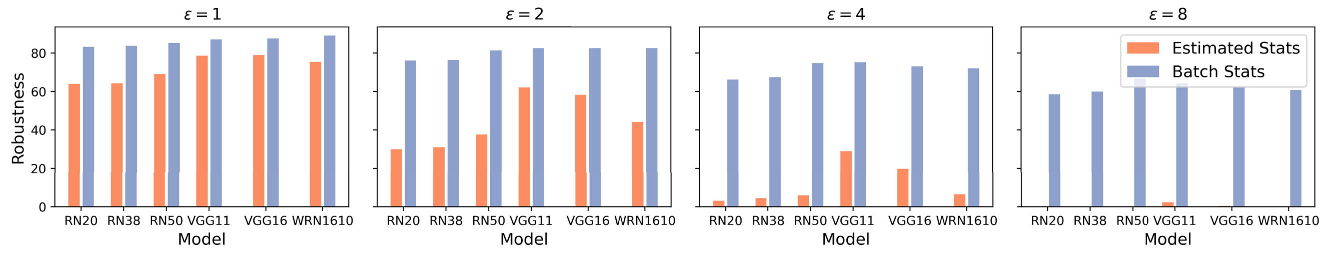


FIGURE 3. Robustness of six different models against query-based blackbox square attack [1]. Robustness of each model is evaluated with batch statistics ($\mu_{\mathcal{B}}, \sigma^2_{\mathcal{B}}$) and population statistics ($\hat{\mu}_{\mathcal{P}}, \hat{\sigma}^2_{\mathcal{P}}$). Robustness of models using batch statistics is much better than models using estimated population statistics.

TABLE 4. {Percentage adversarial accuracy gained compared with baseline when we use batch statistics instead of train-time estimated population statistics. Results are shown for training with clean images (normal training) and training with adversarially perturbed images (adversarial training). As expected, adversarial training shrinks the gap.

Training	FGSM	BIM	MIM	CW	PGD
		CIFAR10			
Normal	39.3{%	102.2{%	101.1{%	71.7{%	101.7{%
Adversarial	23.0{%	30.5{%	31.7{%	57.3{%	30.5{%
		CIFAR100			
Normal	55.0{%	154.8{%	160.0{%	229.5{%	153.4{%
Adversarial	28.6{%	38.1{%	38.5{%	150.0{%	38.1{%

do not suffer from this problem, we repeated Experiment IV-B2 on a query-based BlackBox attack called Square attack [1]. This attack utilizes a random search to find adversarial attacks. Since this attack does not use any gradient information, it can not have the gradient obfuscation issue.

We set the number of queries to 5000 as used by [13]. The robustness of six different models on four different perturbation levels ($\epsilon \in \{1, 2, 4, 8\}$) are reported in Figure 3. Results for each model is reported with population statistics (**P**) and batch statistics (**B**). Replacing population statistics with batch statistics significantly improves square-attack robustness. These results show that our experiments do not suffer from gradient obfuscation problems.

C. SHIFT HYPOTHESIS AND TRANSFER LEARNING

To further test our "Shift Hypothesis", we turn our attention to transfer learning. In transfer learning, we want to transfer the learning of the model trained on a large source dataset to a similar but smaller target dataset. There are many ways to do transfer learning. For instance, using a pre-trained model as a feature extractor and training and classifier on top of it, fine-tuning a few last layers of the pre-trained model on the target dataset, or fine-tuning the whole model on the target dataset [16], [40]. However, the first configuration is of particular interest to us. In this configuration, a model trained on the source dataset is not updated on the target dataset set and uses BatchNorm statistics estimated from the source dataset for feature extraction. We hypothesize that updating BatchNorm's estimate of population statistics on the target dataset can help achieve better accuracy.

To test this, we have used 10 transfer learning datasets listed as follows: Birdsnap [30], Stanford Cars [41], Describable Textures Dataset (DTD) [11], CIFAR10 and 100 [43], UCSD Birds [68], Oxford Flowers [55], Oxford-IIIT Pets dataset [57], Caltech101 [21], Caltech256 [29]. We used Pytorch pre-trained ResNet50[1] as a feature extractor and trained a classifier on top of it without any bells and whistles. To update the BatchNorm layer, we reinitialize the running mean and variance of the BatchNorm layer and calculated them from the target dataset. We also retrained the weight and biases of BatchNorm since they are dependent on sample mean and variance.

We report results in Table and Figure 4. The table shows test accuracy, and the Figure shows the training convergence for these datasets. These results show that we can achieve significant improvement in both accuracy and convergence speed by updating BatchNorm on target datasets. For instance, on the Stanford Cars dataset, the improvement in accuracy is absolute 12%, and on Birdsnap, it is 11.2%. However, an even more interesting observation is the relation of gain of accuracy and the target dataset's similarity with ImageNet. For instance, UCSD-Birds and Birdsnap are birds dataset. However, as shown in the USCD-Birds dataset website,[2] many images of it overlap with ImageNet. This similarity affects the gain of accuracy obtained by updating BatchNorm i.e. improvement for Birdsnap is 10% compare to 2% for USCD-Birds. Similarly, CIFAR10 and 100 are tiny images of size 32 compared to ImageNet's size of 224 and the absolute gain in accuracy is 19% and 23 % respectively. Note that updating BatchNorm always increases accuracy.

V. TOWARDS A ROBUST NORMALIZATION

In Section IV, we empirically show that train-time estimated population statistics of BatchNorm make a deep model more vulnerable to adversarial distribution shift. A straightforward solution - as shown in the experiments (Table 1) - is to use the batch statistics calculated from the inference inputs. However, during training, activations are normalized by the statistics estimated from the large batch. This introduces an intra-batch dependency [35] for BatchNorm. This issue makes BatchNorm dependent on the moving average estimated for inference. In the experiments of the last section, we used a batch size of 128 (same as the training batch size) to validate our hypothesis. However, if we use a small inference

[1]https://pytorch.org/docs/stable/torchvision/models.html
[2]http://www.vision.caltech.edu/visipedia/CUB-200.html

TABLE 5. Role of BatchNorm Statistics for Transfer Learning: Comparison of test accuracy with and without updating BatchNorm statistics on the target dataset. We used a ResNet50 pre-trained on ImageNet as a feature extractor and trained a classifier for the target dataset. For the second case, we also updated BatchNorm statistics on the target dataset.

	Birdsnap	Stanford Cars	DTD	CIFAR 10	CIFAR 100	UCSD Birds	Oxford Flowers	Oxford Pets	Caltech 101	Caltech 256
Classifier w/o Updated BN	43.92	43.08	68.40	67.96	30.53	38.15	94.13	84.30	84.73	76.35
Classifier w/. Updated BN	**55.70**	**66.46**	**70.94**	**86.81**	**53.61**	**39.18**	**96.82**	**86.26**	**89.14**	**77.59**

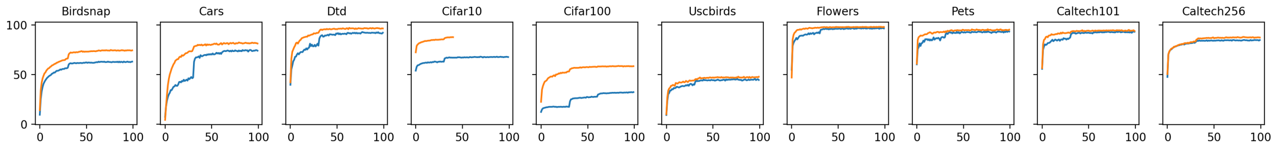


FIGURE 4. Comparison of convergence for training with and without updating BatchNorm for 10 different transfer learning datasets.

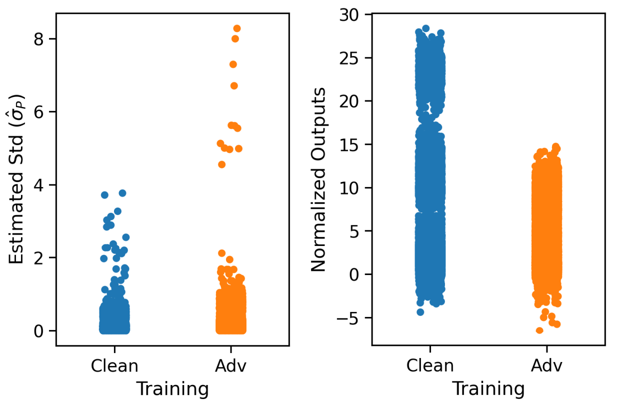


FIGURE 5. (a) Comparison of estimated values of standard deviation $\hat{\sigma}_{\mathcal{P}}$ for ResNet50 trained normally and adversarially on ImageNet. Adversarially trained model has larger values on average as well as has a few outliers. (b) To further see the difference, we visualized the normalization effect of the normally trained and adversarially trained model on unit values. Adversarially trained model limits the output range more harshly compared to a normally trained network.

batch size to calculate statistics, BatchNorm performance descends to zero, as shown in Figure 7. This decrease in performance illustrates that we can not use batch statistics as a solution to this problem.

Recent work on robustness has also shown a connection between the removal of outliers in activations and robustness [18], [74]. Based on these heuristics, min-max normalization becomes a good candidate since it re-scales input (controlling exploding activations), only requires maximum and minimum values, which are not dependent on the distribution, and can remove outliers. We keep using mean considering the importance of centering the data [60]. We define a simplified version of our RobustNorm as:

$$y_i = \frac{x_i - \mu_{\mathcal{B}}}{r_{\mathcal{B}}}, \tag{6}$$

where x_i is i-th example of batch $\mathcal{B}$, range is $r_{\mathcal{B}} = u_{\mathcal{B}} - l_{\mathcal{B}}$, maximum is $u_{\mathcal{B}} = \max_{1 \le i \le M}(x_i)$ and minimum is $l_{\mathcal{B}} = \min_{1 \le i \le M}(x_i)$.

From Von Szokefalvi Nagy inequality ($r_{\mathcal{B}}^2 \le 2n\sigma_{\mathcal{B}}^2$, where n is the number of samples to estimate the range), we can say that range suppresses activations with higher intensity than the variance. BatchNorm uses linear transform to project activations to an appropriate range. However, in our case, the range makes it harder to learn this projection at the start of learning. To make the control more flexible, we introduced a new hyper-parameter – norm power (p). Finally, we define Robust Normalization (RobustNorm or RN) as follows:

$$y_i = \frac{x_i - \mu_{\mathcal{B}}}{r_{\mathcal{B}}^p}. \tag{7}$$

VI. EXPERIMENTS

A. EXPERIMENTAL SETUP

We have used two architectures, ResNet [31] with 20,38 and 50 layers and VGG [65] with 11 and 16 layers. We have used five different datasets for robustness evaluations: MNIST [45], Fashion-MNIST [71], CIFAR10, CIFAR100 [43] and ImageNet [14]. We have always used a learning rate of 0.1 except for no normalization scenarios where convergence is not possible with higher learning rates. In that case, we have used a learning rate of 0.01. We decrease the learning rate ten times at 80th and 120th epoch for CIFAR10, 100; at the 30th epoch for MNIST, Fashion-MNIST; and at 30th, 60th and 90th epoch for ImageNet.

B. EVALUATION ON CIFAR

We evaluated the robustness of RobustNorm for two different datasets. RobustNorm's accuracy is higher in the presence of adversarial attacks (Figure 6). Specifically, RobustNorm increases the adversarial accuracy of ResNet20 from 22% to 70% for CIFAR10 (note that the *epsilon* value is 1/255). All the results are shown in Figure 6.

C. EVALUATION ON CIFAR10 WITH DIFFERENT ARCHITECTURES AND ATTACKS

To further verify the effectiveness of RobustNorm, we trained two models on BatchNorm and RobustNorm. Both models were trained for 100 epochs with an initial learning rate of 0.1 and cosine annealing learning rate decay [47].

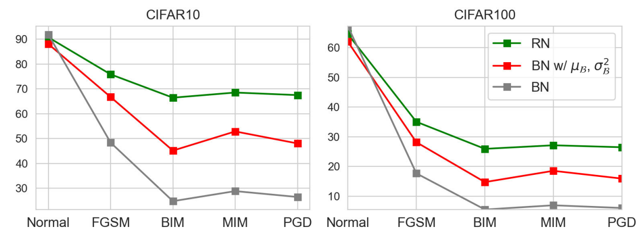


FIGURE 6. Comparison of accuracy of the model with different normalization (RN: RobustNorm, BN: BatchNorm) for ResNet20. In the presence of adversarial attacks, RobustNorm performs better than BatchNorm.

TABLE 6. Robustness of BatchNorm and RobustNorm against four different adversarial attacks. Robustness for each attack is evaluated against four levels of perturbation (ϵ) values. RobustNorm performs well against all of these attacks and perturbation budgets.

Attack	ϵ	ResNet20 BN	ResNet20 RN	VGG16 BN	VGG16 RN
Accuracy	0	92.01	90.03	93.06	92.01
PGD-20	1	43.8	49.63	64.59	64.20
	2	7.67	24.27	27.14	43.59
	4	0.45	10.97	2.91	22.60
	8	0.03	6.17	0	10.13
PGD-100	1	43.3	48.85	64.39	63.82
	2	6.69	21.82	26.53	41.59
	4	0.39	8.9	2.54	18.82
	8	0.02	4.32.	0.07	07.91
APGD-CE	1	12.73	23.43	48.04	48.28
	2	0.22	3.0	9.33	10.4
	4	0	1.29	0.6	0.97
	8	0	1.73	0.0	1.04
Square	1	63.9	84.16	78.95	87.54
	2	29.9	75.9	58.25	80.75
	4	3.15	65.09	19.78	68.25
	8	0.	49.55	0.52	53.33

For RobustNorm, we use $p = 0.2$. We evaluated both of these models on four different attacks: PGD-20, PGD-100, APGD-CE [13] and Square Attack [1]. We used four different perturbation levels e.g., $\epsilon \in \{1, 2, 4, 8\}$ for robustness evaluation. The results are shown in Table 6. RobustNorm performs better than BatchNorm against a diverse set of attacks and perturbation levels.

D. EVALUATION ON ImageNet

To test the effectiveness of RobustNorm at scale, we performed experiments for RobustNorm on ImageNet. Results are shown in Table 7. RobustNorm beats BatchNorm for all the attacks by a wide margin. Note that we have not used any fine-tuning for hyper-parameter p.

E. EVALUATION ON SMALLER BATCH SIZES

RobustNorm performs better compared to BatchNorm when we do not use batch statistics as shown in Figure 7. But, it still suffers some loss of accuracy. Since mean (μ) is a distribution statistic, we use its estimate calculated during training. This improved the performance of RobustNorm for small batch size is shown in Figure 7. To understand the effect of $\hat{\mu}_{\mathcal{P}}$ on adversarial accuracy of RobustNorm, we perform experiments with varying values of ϵ. As shown in Figure 7, adversarial accuracy of RobustNorm with $\hat{\mu}_{\mathcal{P}}$ is comparable to RobustNorm while also having consistent small inference batch performance.

TABLE 7. Comparison of RobustNorm (RN) and BatchNorm (BN) for ImageNet. We have used ResNet18 and $\epsilon = 1/255$. RobustNorm performs better than BatchNorm for all the attacks.

Norm	Normal	FGSM	BIM	MIM	CW	PGD
BN	62.9	12.5	4.0	5.2	3.4	4.4
RN	61.6	28.5	19.8	21.9	22.8	20.5
RN w/ $\hat{\mu}_{\mathcal{P}}$	61.8	28.1	17.7	20.1	25.3	18.7

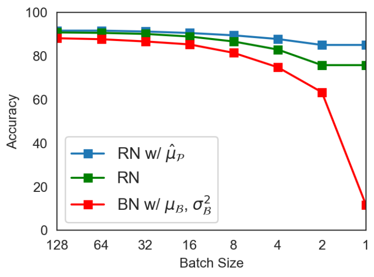


FIGURE 7. Comparison of BatchNorm(BN) and RobustNorm(RN) for small inference batch sizes. RobustNorm performs much better when we only use inference input to compute statistics but it still suffers some loss of accuracy. We achieve performance gain by using the estimated value of the population mean.

F. ABLATION STUDIES

In this section, we have validated and explored different properties of RobustNorm.

1) ANALYSIS OF POWER HYPERPARAMETER

The RobustNorm introduces a new hyperparameter called power or p of the range $r_{\mathcal{B}}$. We found $p = 0.2$ having faster convergence (see Figure (8), red shows $p = 0.2$) and generality across datasets. Therefore, we have used it for all our experiments. Later, we observed that faster convergence does not necessarily mean better adversarial robustness (see Figure 9). For instance, RobustNorm with $p = 0.2$ performs worse in terms of adversarial accuracy when compared to other values. Similarly, $p = 0.05$ has better adversarial robustness in RobustNorm when using the population mean. Therefore, it has room for more improvement by tuning this hyperparameter.

2) SCALIBILITY TO DIFFERENT ARCHITECTURES

We also evaluate RobustNorm to show its scalability on different neural network architectures and depths. We choose ResNet, and VGG architectures as a wide variety of neural networks evolved from these networks. Similarly, VGG was designed before BatchNorm, so it is also interesting to see

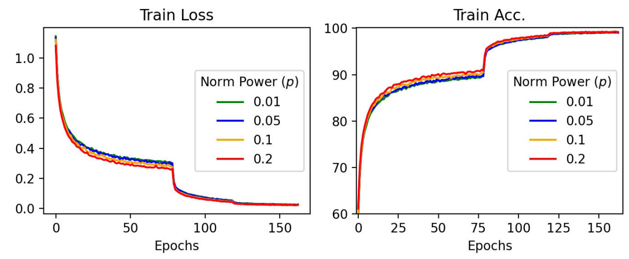


FIGURE 8. Training curves on CIFAR10 dataset for RobustNorm. We compare the loss (left) and accuracy (right) of RobustNorm with different hyperparameter power (p) values. Note that $p = 0.2$ (red line) converges faster than other.

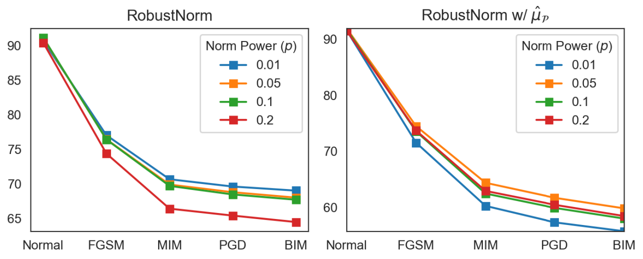


FIGURE 9. The effect of hyperparameter p on adversarial robustness. The results are shown for CIFAR10 and ResNet20.

TABLE 8. Scalibility of RobustNorm across different architectures and depths. BN: BatchNorm, RN: RobustNorm, and RN w/ $\hat{\mu}_{\mathcal{P}}$: RobustNorm with the population mean. Note consistent performance of RobustNorm across architectures and depths.

Model	Norm	FGSM	BIM	MIM	CW	PGD
ResNet38	BN	49.10	18.77	23.90	12.74	20.80
	RN	78.71	71.19	72.74	29.17	72.28
	RN w/ $\hat{\mu}_{\mathcal{P}}$	80.29	71.13	73.23	31.02	72.46
ResNet50	BN	51.14	24.31	29.20	12.53	26.29
	RN	76.58	65.97	68.82	32.19	67.53
	RN w/ $\hat{\mu}_{\mathcal{P}}$	80.10	71.02	72.64	35.76	72.07
VGG11	BN	69.44	61.83	63.70	42.88	62.79
	RN	82.03	77.76	78.94	61.57	78.24
	RN w/ $\hat{\mu}_{\mathcal{P}}$	82.05	76.11	77.88	59.47	76.95
VGG16	BN	63.60	50.15	53.53	30.13	51.52
	RN	83.56	78.75	80.12	50.10	79.28
	RN w/ $\hat{\mu}_{\mathcal{P}}$	82.74	74.73	76.99	49.57	75.81

its performance under different normalizations. To show the scalability of RobustNorm for different depths, we choose two commonly used depths of ResNet (38 and 50) and VGG (11 and 16). Results for the experiments on these architectures for CIFAR10 are shown in Table 8. RobustNorm outperforms BatchNorm by wide margins in all of these networks. For instance, RobustNorm has a margin of 50% with ResNet38, 31% for ResNet50, 15% for VGG11, and 28% for VGG16 when the input has BIM adversarial noise. Similar trends are also visible under different attacks.

VII. CONCLUSION

In this paper, we have investigated the role of BatchNorm in the adversarial vulnerability of convolutional neural networks. We observed that BatchNorm estimates population statistics from natural images during training, and the addition of adversarial noise introduces a targeted distribution shift in the input during inference. We hypothesized that this shift makes train-time estimated statistics inaccurate, thereby contributing to the adversarial vulnerability of BatchNorm. We validated our hypothesis by showing adversarial accuracy differences between statistics calculated from perturbed validation-set batch and train-time estimated statistics. We also showed that normalizations that do not require these train-time estimated values perform better compare with BatchNorm. Based on these insights, we proposed a variant of BatchNorm, which increases the robustness while keeping other benefits of BatchNorm. We have also extended our hypothesis for transfer learning, where we showed that we it is possible to get a significant accuracy gain by updating batch statistics on the target dataset.

REFERENCES

[1] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: A query-efficient black-box adversarial attack via random search," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 484–501.

[2] D. Arpit, Y. Zhou, B. U. Kota, and V. Govindaraju, "Normalization propagation: A parametric technique for removing internal covariate shift in deep networks," 2016, *arXiv:1603.01431*.

[3] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 274–283.

[4] J. Lei Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.

[5] P. Benz, C. Zhang, and I. S. Kweon, "Batch normalization increases adversarial vulnerability and decreases adversarial transferability: A non-robust feature perspective," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 7798–7807.

[6] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," 2017, *arXiv:1712.09665*.

[7] J. Bruce and D. Erman, "A probabilistic approach to systems of parameters and noether normalization," 2016, *arXiv:1604.01704*.

[8] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 39–57.

[9] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin, "On evaluating adversarial robustness," 2019, *arXiv:1902.06705*.

[10] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.

[11] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing textures in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 3606–3613.

[12] T. Cooijmans, N. Ballas, C. Laurent, Ç. Gülçehre, and A. Courville, "Recurrent batch normalization," 2016, *arXiv:1603.09025*.

[13] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *Proc. ICML*, 2020, pp. 2206–2216.

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[15] G. W. Ding, K. Y. C. Lui, X. Jin, L. Wang, and R. Huang, "On the sensitivity of adversarial robustness to input data distributions," in *Proc. ICLR*, 2019, pp. 1–4.

[16] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "DeCAF: A deep convolutional activation feature for generic visual recognition," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 647–655.

[17] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9185–9193.

[18] C. Etmann, S. Lunz, P. Maass, and C.-B. Schönlieb, "On the connection between adversarial robustness and saliency map interpretability," 2019, *arXiv:1905.04172*.

[19] A. Fawzi, S. M. Moosavi-Dezfooli, and P. Frossard, "Robustness of classifiers: From adversarial to random noise," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1632–1640.

[20] A. Fawzi, H. Fawzi, and O. Fawzi, "Adversarial vulnerability for any classifier," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1178–1187.

[21] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories," in *Proc. Conf. Comput. Vis. Pattern Recognit. Workshop*, 2004, p. 178.

[22] N. Ford, J. Gilmer, N. Carlini, and D. Cubuk, "Adversarial examples are a natural consequence of test error in noise," 2019, *arXiv:1901.10513*.

[23] A. Galloway, A. Golubeva, T. Tanay, M. Moussa, and G. W. Taylor, "Batch normalization is a cause of adversarial vulnerability," 2019, *arXiv:1905.02161*.

[24] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2414–2423.

[25] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness," 2018, *arXiv:1811.12231*.

[26] J. Gilmer, L. Metz, F. Faghri, S. S. Schoenholz, M. Raghu, M. Wattenberg, and I. Goodfellow, "Adversarial spheres," 2018, *arXiv:1801.02774*.

[27] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 1, pp. 142–158, Jan. 2016.

[28] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.

[29] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Inst. Technol., Tech. Rep., 2007. [Online]. Available: https://data.caltech.edu/records/nyy15-4j048

[30] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen, "Recent advances in convolutional neural networks," *Pattern Recognit.*, vol. 77, pp. 354–377, May 2018.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[32] L. Huangi, L. Huangi, D. Yang, B. Lang, and J. Deng, "Decorrelated batch normalization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 791–800.

[33] Y. Huang, Y. Cheng, A. Bapna, O. Firat, D. Chen, M. Chen, H. Lee, J. Ngiam, Q. V. Le, and Y. Wu, "GPipe: Efficient training of giant neural networks using pipeline parallelism," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 103–112.

[34] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 125–136.

[35] S. Ioffe, "Batch renormalization: Towards reducing minibatch dependence in batch-normalized models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1945–1953.

[36] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*.

[37] J.-H. Jacobsen, J. Behrmann, R. Zemel, and M. Bethge, "Excessive invariance causes adversarial vulnerability," 2018, *arXiv:1811.00401*.

[38] J.-H. Jacobsen, J. Behrmannn, N. Carlini, F. Tramèr, and N. Papernot, "Exploiting excessive invariance caused by norm-bounded adversarial robustness," 2019, *arXiv:1903.10484*.

[39] J. Kang, S. Tariq, H. Oh, and S. S. Woo, "A survey of deep learning-based object detection methods and datasets for overhead imagery," *IEEE Access*, vol. 10, pp. 20118–20134, 2022.

[40] S. Kornblith, J. Shlens, and Q. V. Le, "Do better ImageNet models transfer better?" in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 2661–2671.

[41] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3D object representations for fine-grained categorization," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, Dec. 2013, pp. 554–561.

[42] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[43] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," California Inst. Technol., Univ. Toronto, Tech. Rep., 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[44] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," 2016, *arXiv:1611.01236*.

[45] Y. LeCun and C. Cortes. (2010). *MNIST Handwritten Digit Database*. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[46] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[47] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," 2016, *arXiv:1608.03983*.

[48] P. Luo, J. Ren, Z. Peng, R. Zhang, and J. Li, "Differentiable Learning-to-Normalize via switchable normalization," 2018, *arXiv:1806.10779*.

[49] P. Luo, P. Zhanglin, S. Wenqi, Z. Ruimao, R. Jiamin, and W. Lingyun, "Differentiable dynamic normalization for learning deep representation," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 4203–4211.

[50] X. Ma, Y. Niu, L. Gu, Y. Wang, Y. Zhao, J. Bailey, and F. Lu, "Understanding adversarial attacks on deep learning based medical image analysis systems," *Pattern Recognit.*, vol. 110, Feb. 2021, Art. no. 107332.

[51] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, *arXiv:1706.06083*.

[52] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3523–3542, Jul. 2022, doi: 10.1109/TPAMI.2021.3059968.

[53] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," 2018, *arXiv:1802.05957*.

[54] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 427–436.

[55] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Proc. 6th Indian Conf. Comput. Vis., Graph. Image Process.*, Dec. 2008, pp. 722–729.

[56] X. Pan, X. Zhan, J. Shi, X. Tang, and P. Luo, "Switchable whitening for deep representation learning," 2019, *arXiv:1904.09739*.

[57] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar, "Cats and dogs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3498–3505.

[58] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[59] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 1–9.

[60] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 901–909.

[61] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Madry, "Adversarially robust generalization requires more data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 5014–5026.

[62] S. Schneider, E. Rusak, L. Eck, O. Bringmann, W. Brendel, and M. Bethge, "Improving robustness against common corruptions by covariate shift adaptation," 2020, *arXiv:2006.16971*.

[63] W. Shao, T. Meng, J. Li, R. Zhang, Y. Li, X. Wang, and P. Luo, "SSN: Learning sparse switchable normalization via SparsestMax," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 443–451.

[64] C.-J. Simon-Gabriel, Y. Ollivier, L. Bottou, B. Schölkopf, and D. Lopez-Paz, "First-order adversarial vulnerability of neural networks and input dimension," 2018, *arXiv:1802.01421*.

[65] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[66] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*.

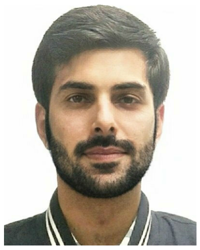
[67] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," 2016, *arXiv:1607.08022*.

[68] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD birds-200–2011 Dataset," California Inst. Technol., Tech. Rep. CNS-TR-2011-001, 2011.

[69] G. Wang, J. Peng, P. Luo, X. Wang, and L. Lin, "Batch Kalman normalization: Towards training deep neural networks with micro-batches," 2018, *arXiv:1802.03133*.

[70] Y. Wu and K. He, "Group normalization," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 3–19.
[71] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.
[72] C. Xie and A. Yuille, "Intriguing properties of adversarial training at scale," 2019, *arXiv:1906.03787*.
[73] C. Xie, M. Tan, B. Gong, J. Wang, A. Yuille, and Q. V. Le, "Adversarial examples improve image recognition," 2019, *arXiv:1911.09665*.
[74] C. Xie, Y. Wu, L. V. D. Maaten, A. L. Yuille, and K. He, "Feature denoising for improving adversarial robustness," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 501–509.
[75] C. Xie, M. Tan, B. Gong, J. Wang, A. L. Yuille, and Q. V. Le, "Adversarial examples improve image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 816–825.
[76] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves ImageNet classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10684–10695.
[77] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv:1605.07146*.

AWAIS MUHAMMAD received the B.S. degree from the University of Engineering and Technology (UET), Lahore, in 2016, and the M.S. degree from Information Technology University (ITU), Lahore, in 2019. He is currently pursuing the Ph.D. degree with the Machine Learning and Visual Computing (MLVC) Laboratory, Kyung Hee University, South Korea. From 2016 to 2019, he was a Research Associate with the Signal Processing and Information Decoding (SPIDER) Laboratory, ITU. From 2020 to 2022, he was an Assistant Researcher with the AI Theory Group, Noah's Ark Laboratory, Huawei, Hong Kong. He is a Research Assistant with Sony AI, Japan. His current research interests include deep learning model architectures, adversarial robustness in the presence of distribution shifts, and transfer learning.



FAHAD SHAMSHAD received the B.S. degree in electrical engineering from the Institute of Space Technology (IST), Islamabad, and the M.S. degree in electrical engineering from the National University of Sciences and Technology (NUST). He is currently a Researcher with the Computer Vision Department, Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi. Before that, he was a Senior Machine Learning Engineer with Pakistani–U.S. AI startup OMNO and a Research Associate with the Center of Artificial Intelligence and Computational Analytics, Information Technology University, Lahore. His research interests include AI security, image restoration, generative models, and optimization theory. He was a recipient of the NUST Merit-Based Scholarship and Pakistan's National Space Agency Scholarship during B.S. and M.S. studies, respectively.

SUNG-HO BAE (Member, IEEE) received the B.S. degree from Kyung Hee University, Republic of Korea, in 2011, and the M.S. and Ph.D. degrees from the Korea Advanced Institute of Science and Technology (KAIST), Republic of Korea, in 2012 and 2016, respectively. From 2016 to 2017, he was a Postdoctoral Associate with the Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. Since 2017, he has been an Assistant Professor with the Department of Computer Science and Engineering (CSE), Kyung Hee University. His current research interests include model compression, interpretation, adversarial attack/defense, and neural architecture search for deep neural networks.