MITIGATING ADVERSARIAL TRAINING INSTABILITY WITH BATCH NORMALIZATION

Arvind P. Sridhar, Chawin Sitawarin, David Wagner [*]
EECS Department, University of California Berkeley
{arvindsridhar,chawins,daw}@berkeley.edu

ABSTRACT

The adversarial training paradigm has become the standard in training deep neural networks for robustness. Yet, it remains unstable, with the mechanisms driving this instability poorly understood. In this study, we discover that this instability is primarily driven by a non-smooth optimization landscape and an internal covariate shift phenomenon, and show that Batch Normalization (BN) can effectively mitigate both these issues. Further, we demonstrate that BN universally improves clean and robust performance across various defenses, datasets, and model types, with greater improvement on more difficult tasks. Finally, we confirm BN's heterogeneous distribution issue with mixed-batch training and propose a solution.

1 INTRODUCTION

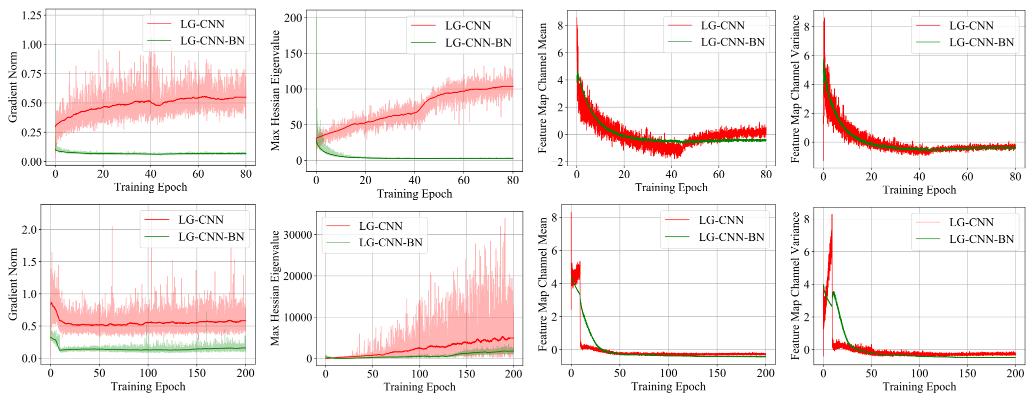
The ability of carefully-crafted adversarial examples to fool deep neural networks (DNNs) has been well documented (Szegedy et al., 2013; Goodfellow et al., 2015; Biggio et al., 2013). In the computer vision domain, adding tiny, imperceptible perturbations to input images can lead state-of-the-art convolutional neural networks (CNNs) to make erroneous predictions with alarming confidence. This phenomenon presents a significant security vulnerability for DNNs deployed in safety-critical settings, as well as highlights the sizeable gap that remains between machine and human intelligence.

The adversarial training (AT) paradigm, first outlined by Madry et al. (2017), has become the standard in training DNNs for robustness to adversarial attacks. The inner maximization problem of AT is usually solved via projected gradient descent (PGD); hence, we refer to this method as PGD-AT. Recently, focus has shifted to certified defenses that provide a guarantee of correct prediction under any allowable perturbation. Interval bound propagation (IBP) presents a loose but computationally efficient upper bound of the adversarial training loss that DNNs can optimize (Gowal et al., 2019).

However, PGD-AT and IBP training strategies remain unstable, with large gradient updates that overcompensate the loss, hurt performance, and, at worst, cause the model to degenerate into always predicting the same class (Gowal et al., 2019). This instability is particularly apparent when training with large perturbation budgets ϵ that increase the difficulty of the task. Various methods have effectively mitigated this instability, e.g., imposing ramp-up curricula to gradually increase adversarial strength for PGD-AT (Sitawarin et al., 2020) and incorporating a tight but more expensive CROWN upper bound to complement IBP (Zhang et al., 2020). However, the underlying mechanisms that drive this instability remain poorly understood, hampering efforts to improve robustness further.

In this study, we diagnose two primary drivers of adversarial training instability that impact both PGD-AT and IBP in comparable fashion: (1) a non-smooth optimization landscape, and (2) an internal covariate shift phenomenon. Furthermore, we discover that Batch Normalization (BN), well-known for improving optimization smoothness and layer independence in the standard setting (Ioffe & Szegedy, 2015; Santurkar et al., 2019), can effectively mitigate these two drivers of instability, preventing model degeneration and leading PGD-AT and IBP to superior optima. To confirm our findings, we comprehensively investigate the specific impact of BN on PGD-AT and IBP performance across multiple datasets, perturbation budgets, and model types. Our results demonstrate that, due to its stabilizing effect, **BN universally improves clean and robust test-time performance for both PGD-AT and IBP, with greater improvement as the perturbation budget becomes larger**.

[*]Please direct correspondence to arvindsridhar@berkeley.edu.



(a) Gradient Norm (b) Max Hessian Eigenval. (c) Feature Map Mean (d) Feature Map Variance

Figure 1: (CIFAR-10) Top row: PGD-AT, bottom row: IBP. (a) Gradient norm and (b) max Hessian eigenvalue indicate smoothness of the loss landscape, and the degree of internal covariate shift is illustrated by the (c) mean and (d) variance of the deepest feature map. For (a) and (b), we overlay the exponential moving average ($\alpha = 0.005$). Metrics are computed at every 10th batch of training.

Nonetheless, when AT is reformulated to jointly optimize both clean and adversarial batches, as in TRADES, we confirm prior findings that BN drives a heterogeneous distribution issue that hurts performance (Xie & Yuille, 2020). We offer a hypothesis for why this issue does not impact IBP, and present a potential solution inspired by IBP's approach. Overall, we demonstrate BN's potency as a stabilizer and smoothing agent for PGD-AT and IBP training, and encourage its use for these two defense strategies; further, we offer a solution for BN's mixed distribution issue that can translate its improvements to strategies such as TRADES. We provide a detailed literature review in Appendix A.

2 Experimental Setup

We use the standard training frameworks for PGD-AT and IBP provided by Madry et al. (2017) and Gowal et al. (2019) respectively. For PGD-AT, we train on the pure adversarial loss, with 40-step PGD for MNIST and 10-step PGD for CIFAR-10. For IBP, we compute bounds using the layer-wise propagation rules derived by Gowal et al. (2019) and Xu et al. (2020), and use a ramp-up period to stabilize training. We experiment with the large CNN model defined by Gowal et al. (2019) (see Table 4) and Pre-Activation ResNet-18 (He et al., 2016). Please see Appendix B for further details.

3 Experimental Results

3.1 Drivers of Adversarial Training Instability

To understand the drivers of instability for PGD-AT and IBP, we visualize several metrics during training: 1) the average norm of the gradients, 2) the maximal eigenvalue of the Hessian, and 3) the distribution of the network's deepest feature map. We present our results, training on CIFAR-10 ($\epsilon_{\text{train}} = 8.8/255$) with the large CNN model, in Figure 1. Both PGD-AT and IBP demonstrate similar trends for all three metrics. The gradient norm and maximal Hessian eigenvalue continue to rise throughout training (red curves), implying that the **optimization landscapes for both PGD-AT and IBP are highly non-smooth**. This corroborates the theoretical analysis performed by Liu et al. (2020), that the loss surface for adversarial training lacks second-order smoothness, contains sharp local minima, and leads to large non-vanishing gradients. Liu et al. (2020) demonstrated these phenomena for PGD-AT; we confirm that IBP's loss surface exhibits similar properties. These phenomena induce large weight updates that overcompensate the loss batch-to-batch, driving instability particularly during the early training epochs. Thus, the model's generalization performance is poor.

Figure 1(c) and (d) plot the change in the distribution of the feature map input to the final convolutional layer of the large CNN as training progresses. We normalize mean and variance values using z-scores to visualize the degree of distributional shift, as opposed to raw shift magnitudes. With both PGD-AT and IBP, the feature map distribution exhibits large shifts (red curves): mean and variance

			LG-CNN (PGD-AT)			LG-CNN-BN (PGD-AT)		
Dataset	ϵ_{train}	ϵ_{test}	Clean	PGD	AA	Clean	PGD	AA
MNIST	0.2	0.1	99.42	98.69	98.55	**99.51**	**98.93**	**98.88**
	0.3	0.2	99.40	97.34	96.91	**99.44**	**97.84**	**97.45**
	0.4	0.3	–	–	–	**99.49**	**97.33**	**94.06**
CIFAR-10	$8.8/255$	$8/255$	65.05	39.20	34.54	**75.52**	**43.22**	**39.44**
	$11/255$	$10/255$	55.78	34.21	28.49	**69.11**	**39.18**	**33.62**
	$17.6/255$	$16/255$	38.89	23.85	16.87	**54.20**	**28.87**	**22.49**

Table 1: PGD-AT performance of large CNN with/without BN for various ϵ on MNIST and CIFAR-10. PGD column represents accuracy under 200-step PGD attack. All values denote accuracies.

oscillate significantly batch-to-batch. This indicates that, contrary to Santurkar et al. (2019)'s findings for clean training, **both PGD-AT and IBP training suffer from an internal covariate shift phenomenon**. Due in part to large weight updates, the distribution of inputs to deep network layers changes constantly, leaving these layers unable to learn a consistent objective and driving instability.

3.2 Batch Normalization Mitigates Instability

Adding BN to the baseline model, **we observe a significant mitigation of both aforementioned drivers of instability and poor generalization for both PGD-AT and IBP**. The BN model exhibits a drastically lower gradient norm and maximal Hessian eigenvalue (green curves), both of which stabilize as training progresses. This demonstrates that BN significantly improves the β-smoothness of the adversarial optimization landscape, leading PGD-AT and IBP to smoother minima with better generalization. Due to smaller weight updates and explicit control of feature map moments, internal covariate shift is limited as well, with the distribution of the network's deepest feature map exhibiting significantly more muted oscillations batch-to-batch. These experiments indicate that BN's known benefits (Ioffe & Szegedy, 2015; Santurkar et al., 2019) extend to the adversarial setting: BN can universally improve loss landscape smoothness and distributional consistency, mitigating instability.

3.3 Empirical Results

We compare BN and non-BN models trained with PGD-AT and IBP to analyze BN's empirical performance benefit. Table 1 presents PGD-AT training results on MNIST and CIFAR-10. In all settings, the large CNN with BN outperforms its non-BN counterpart on both clean accuracy and robustness, reinforcing our analysis that BN effectively mitigates training instability and improves generalization. Notably, for MNIST, the BN model is able to train without degenerating (as opposed to baseline) at $\epsilon_{\text{train}} = 0.4$, scoring **94.06% on the strong Auto-Attack (AA) benchmark** at $\epsilon_{\text{test}} = 0.3$ (the second highest on the leaderboard, with the highest clean accuracy) (Croce & Hein, 2020).

We also present IBP training results on MNIST and CIFAR-10 in Table 2. On MNIST, the IBP-trained BN model achieves **93.38% IBP verified test accuracy, a new state of the art** ($\epsilon_{\text{test}} = 0.3$), outperforming the more expensive CROWN-IBP (Zhang et al., 2020) by 0.4% while maintaining the same clean accuracy. On CIFAR-10, the IBP-trained BN models match the clean and robust performance of CROWN-IBP (trained with non-BN baseline), slightly outperforming at higher ϵ_{train}. These results are significant, considering BN-IBP runs in 80% less time than CROWN-IBP (see B.5).

Larger Improvement for More Difficult Tasks. Tables 1 and 2 demonstrate that, for both PGD-AT and IBP, the BN model's performance improvement over the baseline increases as the task becomes more difficult (with larger ϵ_{train}). In particular, with PGD-AT on CIFAR-10, both clean and robust improvement increase as ϵ_{train} goes up to $17.6/255$; the same is seen with BN-IBP's improvement over CROWN-IBP. As ϵ_{train} increases, training becomes more unstable: the loss landscape becomes sharper and less amenable to optimization (Liu et al., 2020). BN's impact is therefore more pronounced in this setting. By smoothing the loss landscape and limiting degree of internal covariate shift, BN makes optimization significantly more straightforward and drives the model to reach superior local minima. Thus, BN effectively neutralizes the added instability of larger ϵ_{train}.

Heterogeneous Distribution Issue with Mixed Batch Training. Training a Pre-Activation ResNet-18 (PRN-18) model with PGD-AT on CIFAR-10, we observe the same improvement trends

			LG-CNN (IBP)			LG-CNN (CROWN-IBP)			LG-CNN-BN (IBP)		
Dataset	ϵ_{train}	ϵ_{test}	Clean	PGD	IBPV	Clean	PGD	IBPV	Clean	PGD	IBPV
MNIST	0.2	0.1	98.93	98.03	97.26	99.05	98.23	97.62	**99.11**	**98.33**	**97.76**
	0.3	0.2	98.59	96.12	95.38	98.66	96.41	95.99	**98.85**	**97.60**	**96.32**
	0.4	0.3	97.91	93.28	91.25	**98.18**	93.95	92.98	98.17	**96.11**	**93.38**
CIFAR-10	2.2	2	64.24	52.73	43.34	**69.67**	**56.09**	**47.13**	66.89	55.84	46.21
	8.8	8	46.12	31.24	25.26	48.47	33.48	26.50	**48.55**	**34.83**	**27.14**
	11	10	38.59	27.36	19.74	40.05	29.09	23.12	**40.85**	**30.97**	**24.35**

Table 2: IBP performance of large CNN{-BN} on MNIST and CIFAR-10, along with CROWN-IBP (large CNN) to compare. PGD denotes accuracy under 200-step PGD attack. IBPV denotes verified accuracy computed using IBP as the verification method. All CIFAR-10 budgets are divided by 255.

	PRN-18 (no BN)			PRN-18-BN		
Defense	Clean	PGD	AA	Clean	PGD	AA
PGD-AT	55.84	29.77	21.33	**66.44**	**34.03**	**24.26**
TRADES	67.26	**25.84**	**17.34**	**70.42**	24.24	17.18
MAT-100	**81.07**	**24.41**	**16.47**	51.34	23.69	12.46

Table 3: Performance of defenses on CIFAR-10 ($\epsilon_{\text{train}} = \epsilon_{\text{test}} = 16/255$) using Pre-Act ResNet-18 (PGD is 10-step). PGD-AT outperforms with BN, continuing the trend from LG-CNN (Table 1), but mixed distribution issue of BN hurts performance on TRADES and MAT-100 relative to baseline.

due to BN at high $\epsilon_{\text{train}} = 16/255$ (Table 3). Additionally, we train with TRADES (Zhang et al., 2019) and Mixed PGD-AT (MAT-100), which combines both clean and adversarial cross-entropy losses to optimize both objectives simultaneously (Xie et al., 2019). Specifically, when training with MAT-100, Xie & Yuille (2020) noted a heterogeneous distribution issue with BN-infused ResNet models at large ϵ_{train} on ImageNet. Because clean and adversarial batches follow two distinct distributions, normalizing both with a single BN layer results in BN attempting to model a mixture distribution, hurting robustness significantly compared to pure PGD-AT. We confirm that this issue exists with CIFAR-10 as well, when both clean and adversarial batches are trained on simultaneously (TRADES, MAT-100). With TRADES ($\beta = 6$), BN shows little impact on performance, in contrast to PGD-AT. With MAT-100, PRN-18-BN significantly underperforms both the baseline (no BN) and PGD-AT on both clean and robust metrics, reinforcing the analysis of Xie & Yuille (2020).

4 Conclusions and Future Directions

In this study, we diagnose two primary drivers of instability during adversarial training and confirm their analogous impact for both PGD-AT and IBP: a non-smooth optimization landscape and an internal covariate shift phenomenon. To this end, we demonstrate that Batch Normalization effectively mitigates both these issues and acts as a significant stabilizing influence during adversarial training, enabling better clean and robust generalization. Our conclusions are validated by empirical analysis on MNIST and CIFAR-10 datasets with both conventional and residual networks, where we show that BN universally improves performance across PGD-AT and IBP formulations. Further, we demonstrate that BN's performance improvement increases as the task becomes more difficult.

In addition, we confirm prior analysis that, when AT is reformulated to train on both clean and adversarial batches, BN drives a heterogeneous distribution issue that hurts performance. We note that IBP, which also jointly optimizes clean and robust loss, does not suffer from this mixed distribution issue. The BN propagation rule for IBP (Alg. 1) specifies that, during training, the clean batch's moments are used to normalize the upper/lower bounds and update BN's running state. This consistency enables the model to learn how to correctly classify adversarial examples normalized with clean moments; thus, at test time, robustness is maintained. We hypothesize that this approach can translate to TRADES and MAT-100. Namely, at train time, use the moments of either the clean or adversarial batch to normalize both batches and update BN's state. The model learns how to optimize both tasks using the single normalization basis, providing consistency between train and test time and avoiding mixed distributions. We look forward to testing this hypothesis in future research.

Acknowledgements

This paper was supported by the Hewlett Foundation through the Center for Long-Term Cybersecurity, the Berkeley Deep Drive project, and gifts from Open Philanthropy, Google, and Futurewei.

References

Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *CoRR*, abs/1802.00420, 2018. URL `http://arxiv.org/abs/1802.00420`.

Muhammad Awais, Fahad Shamshad, and Sung-Ho Bae. Towards an adversarially robust normalization approach. *arXiv:2006.11007 [cs, stat]*, June 2020.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, 2009.

Philipp Benz, Chaoning Zhang, and In So Kweon. Batch normalization increases adversarial vulnerability: Disentangling usefulness and robustness of model features. *arXiv:2010.03316 [cs, stat]*, October 2020.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Železný (eds.), *Machine Learning and Knowledge Discovery in Databases*, pp. 387–402, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-40994-3.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12): 124018, 2019.

Minhao Cheng, Qi Lei, Pin-Yu Chen, Inderjit S. Dhillon, and Cho-Jui Hsieh. Cat: Customized adversarial training for improved robustness. *CoRR*, abs/2002.06789, 2020. URL `http://arxiv.org/abs/2002.06789`.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020.

Gavin Weiguang Ding, Yash Sharma, Kry Yik Chau Lui, and Ruitong Huang. Mma training: Direct input space margin maximization through adversarial training. In *ICLR*, 2020. URL `https://openreview.net/forum?id=HkeryxBtPB`.

Angus Galloway, Anna Golubeva, Thomas Tanay, Medhat Moussa, and Graham W. Taylor. Batch normalization is a cause of adversarial vulnerability. *arXiv:1905.02161 [cs, stat]*, May 2019.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv:1810.12715 [cs, stat]*, August 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pp. 630–645. Springer, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456. PMLR, 2015.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

J. Zico Kolter and Eric Wong. Provable defenses against adversarial examples via the convex outer adversarial polytope. *CoRR*, abs/1711.00851, 2017. URL `http://arxiv.org/abs/1711.00851`.

Chen Liu, Mathieu Salzmann, Tao Lin, Ryota Tomioka, and Sabine Süsstrunk. On the loss landscape of adversarial training: Identifying challenges and how to overcome them. In *Advances in Neural Information Processing Systems*, 2020.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *CoRR*, abs/1706.06083, 2017. URL `http://arxiv.org/abs/1706.06083`.

Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study, 2018.

Leslie Rice, Eric Wong, and J. Zico Kolter. Overfitting in adversarially robust deep learning, 2020.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *arXiv:1805.11604 [cs, stat]*, April 2019.

Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *Advances in Neural Information Processing Systems 32*. 2019.

Chawin Sitawarin, Supriyo Chakraborty, and David Wagner. Improving adversarial robustness through progressive hardening. *arXiv:2003.09347 [cs, stat]*, June 2020.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013. URL `http://arxiv.org/abs/1312.6199`.

Yisen Wang, Xingjun Ma, James Bailey, Jinfeng Yi, Bowen Zhou, and Quanquan Gu. On the convergence and robustness of adversarial training. In *ICML*, 2019.

Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *ICLR*, 2020. URL `https://openreview.net/forum?id=rklOg6EFwS`.

Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *ICLR*, 2020. URL `https://openreview.net/forum?id=BJx040EFvH`.

Yuxin Wu and Kaiming He. Group normalization. *CoRR*, abs/1803.08494, 2018. URL `http://arxiv.org/abs/1803.08494`.

Cihang Xie and Alan Yuille. Intriguing properties of adversarial training at scale. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HyxJhCEFDS`.

Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L. Yuille, and Quoc V. Le. Adversarial examples improve image recognition. *CoRR*, abs/1911.09665, 2019. URL `http://arxiv.org/abs/1911.09665`.

Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond, 2020.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. *CoRR*, abs/1901.08573, 2019. URL `http://arxiv.org/abs/1901.08573`.

Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in Neural Information Processing Systems (NuerIPS)*, December 2018.

Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. In *International Conference on Learning Representations*, 2020.

A Related Work

A.1 Adversarial Examples

Adversarial examples in the computer vision domain are generated by adding a small perturbation δ to an input image x (Biggio et al., 2013; Goodfellow et al., 2015; Szegedy et al., 2013). δ is typically constrained to be within some ℓ_p-norm ball with a radius of ϵ, ensuring that the perturbed image $x + \delta$ is perceptually "close" to x. To compute the optimal perturbation δ^*, the following optimization problem is solved:

$$x_{\text{adv}} = x + \delta^* \quad \text{where} \quad \delta^* = \underset{\delta : \|\delta\|_p \leq \epsilon}{\arg\max} \; L(x + \delta) \tag{1}$$

Here, $L : \mathbb{R}^d \to \mathbb{R}$ is the loss function of the target neural network. Projected gradient descent (PGD) is often used to iteratively optimize Eqn. 1 in the white-box setting, where the attacker has access to the model's parameters (Madry et al., 2017). In this study, we focus on ℓ_∞-norm balls.

A.2 Adversarial Training

The popular adversarial training (AT) paradigm developed by Madry et al. (2017) involves generating $\ell_p(\epsilon)$-constrained loss-maximizing adversarial examples from each batch of training data via Eqn. 1, computing the expected adversarial loss over all perturbed samples x_{adv} in the batch, and training the model to minimize this loss. Mathematically, AT formulates the following saddle point optimization problem, where θ are the parameters of the model and $\{(x_i, y_i)\}_{i=1}^n$ the training set:

$$\underset{\theta}{\arg\min} \; \frac{1}{n} \sum_{i=1}^{n} L_\epsilon(x_i; \theta) \tag{2}$$

$$\text{where} \quad L_\epsilon(x; \theta) = \max_{\delta : \|\delta\|_p \leq \epsilon} L(x + \delta; \theta) \tag{3}$$

Madry et al. (2017) used multi-step PGD to compute adversarial examples (inner maximization) and standard optimizers, e.g., SGD or Adam, to train the network (outer minimization). Defining L to be the cross-entropy loss, we denote this technique as PGD-AT. Since then, several works have sought to improve PGD-AT, by reducing computation time (Shafahi et al., 2019; Wong et al., 2020), trying different loss functions to improve robustness (Zhang et al., 2019; Ding et al., 2020; Wang et al., 2020), and imposing a curriculum to gradually ramp-up the difficulty of computed adversarial examples and stabilize large-ϵ training (Sitawarin et al., 2020; Wang et al., 2019; Cheng et al., 2020).

In particular, TRADES (Zhang et al., 2019) maximized the KL divergence between clean and adversarial logits, as opposed to the adversarial cross-entropy loss, during PGD (inner maximization). Their training loss (outer minimization) is the following joint formulation of clean and robust loss:

$$L_{\text{TRADES}} = L_{CE}(x_{\text{clean}}; y; \theta) + \beta L_{KL}(x_{\text{clean}}; x_{\text{adv}}; \theta) \tag{4}$$

Here, L_{CE} denotes the cross-entropy loss, L_{KL} the KL divergence (between clean and adversarial logits), x_{adv} the KL divergence-maximizing adversarial example found via PGD, and β the robust loss weight.

We also define Mixed PGD-AT (MAT-100), a joint training strategy derived from PGD-AT to optimize for both clean accuracy and robustness in an equal ratio (100% clean + 100% adversarial):

$$L_{\text{MAT-100}} = L_{CE}(x_{\text{clean}}; y; \theta) + L_{CE}(x_{\text{adv}}; y; \theta) \tag{5}$$

We can also pre-multiply the clean cross-entropy by a weight β, e.g. $\beta = 0.2$ to produce MAT-20.

A.3 Certified Defenses

Models trained with PGD-AT, while robust under strong empirical attacks (Croce & Hein, 2020; Athalye et al., 2018), lack robustness guarantees. This is due to the nonconvexity of neural networks: PGD can only compute a lower bound of the maximal robust loss $L_\epsilon(x; \theta)$ for each sample, and minimizing a lower bound does not guarantee that Eqn. 2 is indeed minimized (Zhang et al., 2020). Specifically, there could be a valid perturbation δ that PGD does not find, but $x + \delta$ still incurs a high loss that is never optimized. Certified defenses seek to address this concern, using a neural network verification method to instead compute an upper bound of $L_\epsilon(x;\theta)$ (Kolter & Wong, 2017). Minimizing an upper bound of worst-case robust loss guarantees that Eqn. 2 is minimized, ensuring that the network's prediction is correct for all valid perturbations within the norm ball. To compute this upper bound, Kolter & Wong (2017) proposed linearly relaxing the activations of the target network and propagating bounds through this convex adversarial polytope (CAP); this technique was further improved by CROWN (Zhang et al., 2018). While the resulting upper bound is tight, computing this bound is quite computationally expensive, the model's clean accuracy tends to suffer, and the network tends to be over-regularized during training (Zhang et al., 2020).

A.4 Interval Bound Propagation

Interval bound propagation (IBP), developed by Gowal et al. (2019), uses a simple propagation rule to derive an upper bound for robust loss $L_\epsilon(x;\theta)$ in computation time equivalent to just two forward passes through the network, yielding an efficient certified defense. The resulting bound is much looser compared to other certificates, since activations are not linearized and correlations between neurons of different layers are not considered. Thus, IBP training tends to be unstable and hard to tune, especially during the initial epochs. Some of this instability can be mitigated using a ramp-up curriculum that gradually increases ϵ_{train} and the adversarial exposure of training loss, enabling IBP to achieve state-of-the-art certified robustness with only limited sacrifice in clean accuracy (Gowal et al., 2019). CROWN-IBP (Zhang et al., 2020) additionally incorporated a CROWN upper bound into the loss formulation during the initial epochs of training, tightening the initial bound and stabilizing training. When used in conjunction with a ramp-up curriculum, CROWN-IBP sets the state of the art in clean and certified robust accuracy at various ϵ_{test} on MNIST and CIFAR-10, consistently outperforming IBP. However, computing the CROWN bound remains computationally expensive.

The IBP training loss uses a joint formulation with both clean and robust (upper-bounded) loss:

$$L_{\text{IBP}} = \kappa L_{CE}(x;y;\theta) + (1-\kappa)L_{CE}(-\underline{m}_{\text{IBP}}(x,\epsilon);y;\theta) \tag{6}$$

Here, $\underline{m}_{\text{IBP}}$ denotes the worst-case adversarial logit vector, using the lower bound logit for class y_i that x belongs to and the upper bound logit for all other y_j. Thus, the second term is an upper bound of robust loss. IBP typically elides the last linear layer in bound propagation to improve tightness.

A.5 Instability during Adversarial Training

The instability of adversarial training methods, particularly with large ϵ_{train}, has been well documented (Zhang et al., 2020; Sitawarin et al., 2020; Liu et al., 2020). With these more difficult tasks, training with vanilla PGD-AT and IBP (without curriculum) causes the model to degenerate into always predicting the same class. Even with smaller perturbation budgets, training can be unstable, particularly during the initial epochs of training – when large gradients and a high learning rate induce the model to make large weight updates that tend to overcompensate adversarial loss.

Liu et al. (2020) studied the loss surface of PGD-AT to understand the mechanisms driving this instability and found that larger adversarial budgets ϵ lead to sharper minima in the loss landscape (and less smooth loss surface overall). This phenomenon, termed *gradient scattering*, causes large non-vanishing gradients to persist in the final stages of training and leaves the model unable to reach the best local minima. Specifically, Liu et al. (2020) mathematically demonstrated that the adversarial objective function lacks second-order smoothness, meaning that gradients in arbitrarily small neighborhoods in θ-space can change discontinuously (more so with large ϵ). They also demonstrated that SGD on the adversarial loss is not guaranteed to converge to a critical point. These findings reinforce the empirical and theoretical evidence suggesting that smooth loss landscapes, particularly around local minima, speed up convergence of SGD and improve the network's generalization (Chaudhari et al., 2019; Keskar et al., 2016; Novak et al., 2018). Overall, the sharp adversarial loss landscape of PGD-AT is not favorable to optimization, and models trained with PGD-AT suffer from overfitting and are not able to generalize as well as clean-trained models (Rice et al., 2020). Whether these same properties hold true for IBP certified defense training is yet to be studied.

Model size also plays a significant role in the stability of AT. Xie & Yuille (2020) found that PGD-AT requires substantially larger model capacity in order to converge. In general, larger and deeper networks consistently boost robustness; sustained improvement is observed even with models as deep as ResNet-638. Digging deeper, Liu et al. (2020) showed that smaller and shallower models trained with PGD-AT, especially with larger ϵ, are more likely to have dead layers, i.e. layers in which all activated neuron outputs are zeros. This causes model degeneration, since prior layers no longer train and later layers can at best learn to predict a constant output. IBP and CROWN-IBP were also shown to perform better with larger models, presumably for the same reasons as PGD-AT.

Curriculum learning, which gradually ramps up the difficulty of the training task, has been shown to mitigate PGD-AT and IBP instability, improve the quality of found local minima, and significantly boost robustness (Zhang et al., 2020; Sitawarin et al., 2020; Liu et al., 2020). As Bengio et al. (2009) describes, a curriculum tends to smooth the objective function, resulting in a loss landscape that is more easily optimizable. Intuitively, because ramp-up curricula in AT manipulate ϵ either directly or indirectly, they can explicitly control the smoothness of the loss landscape at any point in time, enabling them to bias the trajectory of SGD towards smoother minima. The empirical results validate these theoretical claims, with curriculum-mediated PGD-AT and IBP algorithms significantly outperforming the vanilla methods in both clean and robust test accuracy, while also being less sensitive to initialization and choice of learning rate (Zhang et al., 2020; Liu et al., 2020).

A.6 Batch Normalization in Adversarial Training

Given a batch $\{x_i\}_{i=1}^{\mathcal{B}}$, Batch Normalization performs the following normalization on each x_i:

$$BN(x_i) = \left(\frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}\right)\gamma + \beta \tag{7}$$

Here, $\mu_{\mathcal{B}}$ and $\sigma_{\mathcal{B}}^2$ denote the channel-wise mean and variance of the input batch $\{x_i\}_{i=1}^{\mathcal{B}}$. γ and β are the trainable scale and shift parameters, respectively. At test time, the concept of a "batch" disappears; thus, BN uses running statistics $\mu_{\mathcal{R}}$ and $\sigma_{\mathcal{R}}^2$, computed during training as an exponential moving average of the training batch moments, to normalize each x_i.

In the natural setting, BN (Ioffe & Szegedy, 2015) has become a crucial component of modern state-of-the-art computer vision models, helping them train faster and optimize more effectively. BN's improvement was initially attributed to it reducing internal covariate shift (ICS) among the inputs to different layers of the network. The ICS thesis postulates that as models train and their weights change, the distribution of feature maps inputs to deep layers in the network continuously shifts; as a result, these deeper layers fruitlessly endeavor to optimize an objective that, from their perspective, is constantly changing. BN mitigates ICS by standardizing the output distribution of each layer, limiting the degree of shift batch-to-batch and enabling later layers to learn independently of changes in earlier layers. BN also has regularizing effects on the network due to the added noise of using batch statistics, improving generalization (Ioffe & Szegedy, 2015).

Digging deeper, Santurkar et al. (2019) found that, more than mitigating ICS, BN increases the β-smoothness of the optimization landscape, improving gradient predictiveness and enabling faster and more direct convergence to local minima. Specifically, BN improves the Lipschitzness of the loss function, meaning that the loss changes at a smaller rate and gradients are smaller in magnitude. This enables more stable training that is less sensitive to choice of learning rate and weight initialization. To analyze Lipschitzness and β-smoothness, Santurkar et al. (2019) measured the absolute and relative changes in gradient magnitude along the gradient direction as training progressed.

In our study, we perform the same analysis as Santurkar et al. (2019) in the adversarial setting. We confirm that BN's β-smoothness improvements extend to PGD-AT and IBP. However, contrary to Santurkar et al. (2019), we do in fact observe a non-negligible ICS phenomenon with non-BN adversarial training; further, we find that BN effectively mitigates this ICS (Figure 1). Since AT involves a significantly more intensive and dynamic training curriculum compared to standard training, we hypothesize that batch-to-batch distributional shifts become more pronounced in this setting, compounding the adverse impacts of the sharp optimization landscape and driving further instability.

In the adversarial setting, conclusions regarding BN's impact on robustness are mixed. Xu et al. (2020) found that a seven-layered CNN model with BN after each convolution was able to outperform its non-BN counterpart as well as more complex DenseNet, WideResNet, and ResNeXt architectures when trained with CROWN-IBP on CIFAR-10 ($\epsilon_{\text{test}} = 8/255$).

By contrast, Xie & Yuille (2020) found that training a BN-infused ResNet model on ImageNet ($\epsilon_{\text{train}} = 16/255$) with MAT-100, which jointly optimizes clean and adversarial cross-entropy loss in the same ratio, drove a heterogeneous distribution issue with BN. Because clean and adversarial images are drawn from two different domains, it is challenging for BN to estimate normalization statistics (i.e. $\mu_{\mathcal{R}}$ and $\sigma^2_{\mathcal{R}}$) for this mixture distribution if both tasks are optimized concurrently. Xie & Yuille (2020) showed that clean and adversarial input images induce significantly different feature map moment statistics at each layer, particularly for the large ϵ on the complex ImageNet dataset. Thus, using a single BN layer led to poor, non-asymptotic robustness compared to pure PGD-AT.

To solve this issue, Xie & Yuille (2020) proposed MBN, effectively de-coupling the distribution by training separate BN layers for clean and adversarial batches. At test time, they used one set of BNs to normalize both clean and adversarial inputs. They showed that using MBN_{adv} led to asymptotic robustness on par with pure PGD-AT, and using $\text{MBN}_{\text{clean}}$ led to clean accuracy on par with pure clean training (Xie et al., 2019). They also suggested using batch-unrelated normalization layers, such as Group Normalization (Wu & He, 2018), to avoid the issue of running statistics altogether.

Other studies have echoed the same findings regarding the mixed distribution issue of BN with large perturbation budgets (Awais et al., 2020; Benz et al., 2020; Galloway et al., 2019). However, to our knowledge, **no prior work has systematically studied the impact of BN across a variety of adversarial training strategies, datasets, budgets, and model types, to identify in which cases BN helps and in which cases it hurts. We hope to fulfill this role to aid future work in this field.**

B Training Details

We provide detailed descriptions of our models and training frameworks below.

B.1 Models Used

For both PGD-AT and IBP, we train with the large CNN model defined by Gowal et al. (2019) (their Table 1), a conventional CNN architecture with ReLU nonlinearities, 5 convolutional layers, and 1 fully-connected layer. This model has 17 million trainable parameters. We optimize using Adam optimizer with default parameters in PyTorch, following Gowal et al. (2019). We do not use weight decay. We train with a batch size of 100 in all cases. We also define small and medium CNNs of the same flavor, with fewer convolutional layers. Detailed architectures of all 3 models are in Table 4.

Small CNN	Medium CNN	Large CNN
Conv 16 4×4+2	Conv 32 3×3+1	Conv 64 3×3+1
Conv 32 4×4+1	Conv 32 4×4+2	Conv 64 3×3+1
FC 100	Conv 64 3×3+1	Conv 128 3×3+2
	Conv 64 4×4+2	Conv 128 3×3+1
	FC 512	Conv 128 3×3+1
	FC 512	FC 512
# params: 471K	1.2M	17M

Table 4: Convolutional model architectures used for PGD-AT and IBP training. All convolutional and fully-connected layers are followed by ReLU activations. Conv k w×h+s denotes a 2D convolutional layer with k filters of size w×h using stride s in both dimensions. FC n denotes a fully-connected layer with n outputs. The last row contains number of trainable parameters for CIFAR-10.

We abbreviate these models as SM-CNN, MD-CNN, and LG-CNN respectively. To construct LG-CNN-BN, we add a 2D Batch Normalization layer after each convolutional layer (prior to ReLU), and a 1D Batch Normalization layer after each fully-connected layer (also prior to ReLU).

For PGD-AT on CIFAR-10, we additionally train with a Pre-Activation ResNet-18 model (He et al., 2016). We denote PRN-18 as the model with all Batch Normalization layers eliminated, and PRN-18-BN as the standard ResNet model with Batch Normalization within each PreAct block. We train with a batch size of 128 using SGD optimizer with momentum of 0.9 and weight decay of 5×10^{-4}.

B.2 PGD-AT Training Framework

We use untargeted projected gradient descent (PGD) with uniform random initialization within the ϵ-bounded ℓ_∞-ball and 1 random start (no restarts) to generate adversarial examples at training time. We train the network on the pure adversarial cross-entropy loss, as outlined by Madry et al. (2017).

MNIST. We train for 50 epochs using LG-CNN with/without BN. We use learning rate of 1×10^{-3}, decreased by factor of 10 at epochs 30 and 40. We train with $\epsilon_{\text{train}} = 0.1 + \epsilon_{\text{test}}$ in order to maximize robustness at the target perturbation budget. We train with 40-step PGD in all cases. For $\epsilon_{\text{train}} = 0.2$, we use PGD step size $\alpha = 0.01$. For $\epsilon_{\text{train}} = 0.3$ and $\epsilon_{\text{train}} = 0.4$, we use PGD step size $\alpha = 0.02$.

CIFAR-10. We train for 80 epochs. For LG-CNN, we use learning rate of 1×10^{-3}, decreased by factor of 10 at epochs 44 and 66. We train with $\epsilon_{\text{train}} = 1.1 \times \epsilon_{\text{test}}$ for LG-CNN in order to maximize robustness at the target perturbation budget. For PRN-18, we use learning rate of 5×10^{-2}, decreased by factor of 10 at epochs 40, 55, and 70. We train with $\epsilon_{\text{train}} = \epsilon_{\text{test}}$ in this case so as to follow prior convention for this model (Sitawarin et al., 2020). We train with 10-step PGD in all cases. For $\epsilon_{\text{train}} = 8.8/255$, we use PGD step size $\alpha = 2/255$. For $\epsilon_{\text{train}} = 11/255$, we use PGD step size $\alpha = 3/255$. For $\epsilon_{\text{train}} = 17.6/255$, we use PGD step size $\alpha = 4/255$. We perform standard pre-processing and data augmentation: random crop, flip, and input normalization with dataset statistics.

B.3 IBP Training Framework

We use LG-CNN with and without BN in all cases, and use the same training framework for MNIST and CIFAR-10. We train for 200 epochs with learning rate of 5×10^{-4}, decreased by factor of 10 at epochs 130 and 190. The first 10 epochs are clean training, with $\epsilon_{\text{train}} = 0$ and $\kappa = 1$, where κ denotes the weight of the clean cross-entropy loss and $1 - \kappa$ the weight of the robust loss in the IBP loss formulation (Eqn. 6). Then, from epochs 10 to 60, we linearly decrease κ to 0.5 and increase ϵ_{train} to $\epsilon_{\text{train-final}}$. We train with $\epsilon_{\text{train}} = 0.1 + \epsilon_{\text{test}}$ for MNIST and $\epsilon_{\text{train}} = 1.1 \times \epsilon_{\text{test}}$ for CIFAR-10 following prior convention for IBP (Zhang et al., 2020). We perform the same standard pre-processing and data augmentation techniques (crop, flip, input normalization) for CIFAR-10. We also train with CROWN-IBP with same framework using code provided by Zhang et al. (2020).

B.4 Other Frameworks

For TRADES, we use the same training framework as PGD-AT with the PRN-18 model on CIFAR-10. We use $\beta = 6$ as the weight for the robust loss in the TRADES loss formulation (Eqn. 4). For MAT-100, we again use the same training framework as PGD-AT with PRN-18 on CIFAR-10.

For all training runs, we perform early stopping, in that we only save the model with the best sum of clean and robust accuracy on a hold-out validation set (separately given for MNIST, randomly chosen from 10% of training samples for CIFAR-10). We then evaluate this best model on the test set, corresponding to our results in Tables 1, 2, and 3. For Auto-Attack (Croce & Hein, 2020), we use the standard attack mechanism (APGD-CE, APGD-T, FAB-T, and Square) with default parameters.

In Figure 1, we compute all metrics with every 10th training batch. For the gradient norm, we compute the average norm of all gradients at that batch, plotting the result with its exponential moving average. We also compute the maximum eigenvalue of the Hessian matrix at this batch, using the PyHessian package. For the internal covariate shift studies, we compute the channel-wise feature map mean and variance across the entire batch, and average the moments across all channels. We then standardize each metric curve using z-scores. Specifically, we compute the average value of all points on that curve, as well as the standard deviation of the points, and standardize via $(x - \mu)/\sigma$ for all points on the curve. This enables us to gauge the degree of batch-to-batch shift in these feature map moments on a percentage basis, providing a fair comparison for internal covariate shift analysis.

B.5 Training Infrastructure and Runtimes

All of our code is written in PyTorch, and we train all our models on a server with 1 NVIDIA Tesla V100 GPU, 2 CPU cores, and 13 GB memory. With this setup, PGD-AT training with LG-CNN{-BN} takes about 45 minutes on MNIST and 1 hour on CIFAR-10. PGD-AT with PRN-18{-BN} on CIFAR-10 takes about 3 hours, MAT-100 takes 3.5 hours, and TRADES takes 4 hours.

For IBP with LG-CNN{-BN}, MNIST takes about 1 hour and CIFAR-10 takes about 1.2 hours. By contrast, CROWN-IBP with LG-CNN, using the publicly-available code from Zhang et al. (2020), takes 4.6 hours on MNIST and 5.4 hours on CIFAR-10 with the same training framework and infrastructure setup. Thus, having LG-CNN-BN trained with IBP match the performance of, and at times outperform, CROWN-IBP (Table 2) in a fraction of the training time is a significant result.

C Batch Normalization Propagation Rule

We outline the IBP propagation rule for Batch Normalization in Algorithm 1. This is the same rule defined by Xu et al. (2020) to propagate the clean map as well as upper/lower bounds through BN.

Algorithm 1: IBP Propagation Rule for BN

```
Input      : Feature map and bounds $z^{(l)},\ \underline{z}^{(l)},\ \overline{z}^{(l)}$
Output     : Normalized feature map and bounds, in the same order as input
Parameters: Layer parameters $\gamma,\ \beta,\ \mu_{\mathcal{R}},\ \sigma^2_{\mathcal{R}}$
1  $\mu \leftarrow$ batch-mean$(z^{(l)})$ if train-mode else $\mu_{\mathcal{R}}$;
2  $\sigma^2 \leftarrow$ batch-variance$(z^{(l)})$ if train-mode else $\sigma^2_{\mathcal{R}}$;
3  $\widehat{\underline{z}^{(l)}} \leftarrow (\underline{z}^{(l)} - \mu)/\sqrt{\sigma^2 + \epsilon_{\text{layer}}}$;
4  $\widehat{\overline{z}^{(l)}} \leftarrow (\overline{z}^{(l)} - \mu)/\sqrt{\sigma^2 + \epsilon_{\text{layer}}}$;
5  $m_p \leftarrow \gamma\big(\widehat{\overline{z}^{(l)}} + \widehat{\underline{z}^{(l)}}\big)/2 + \beta$;
6  $d_p \leftarrow |\gamma|\big(\widehat{\overline{z}^{(l)}} - \widehat{\underline{z}^{(l)}}\big)/2$;
7  $\widehat{z^{(l)}} \leftarrow \gamma\big[(z^{(l)} - \mu)/\sqrt{\sigma^2 + \epsilon_{\text{layer}}}\big] + \beta$;
8  if train-mode then
9      update-running-statistics$(\mu,\ \sigma^2,\ \text{momentum})$;
10 end
11 return $\widehat{z^{(l)}},\ (m_p - d_p),\ (m_p + d_p)$;
```

D Additional Results

D.1 Investigating Model Size and Batch Normalization Impact: PGD-AT

We investigate the differences in the impact of BN with the small, medium, and large CNN models defined in Table 4. For PGD-AT, we experiment with MNIST at $\epsilon_{\text{train}} = 0.3$ and $\epsilon_{\text{test}} = 0.2$; our results are in Table 5. Intriguingly, BN's improvement is largest for the small CNN model, causing robustness to jump over 4%. The medium CNN exhibits the next largest improvement, while the large CNN exhibits the smallest relative improvement. Intuitively, smaller models are more sensitive to smoothness of loss surface, as a non-smooth loss surface can quickly lead to degeneration (Liu et al., 2020). As a result, BN's stabilizing influence is most acutely felt at the small CNN level.

	No BN (PGD-AT)		With BN (PGD-AT)	
Model	Clean	PGD	Clean	PGD
SM-CNN	97.51	92.90	**99.30**	**97.04**
MD-CNN	99.32	97.32	**99.43**	**97.76**
LG-CNN	99.40	97.34	**99.44**	**97.84**

Table 5: PGD-AT performance of various model sizes with/without BN on MNIST ($\epsilon_{\text{train}} = 0.3$, $\epsilon_{\text{test}} = 0.2$). PGD column represents accuracy under 200-step PGD attack.

D.2 Investigating Model Size and Batch Normalization Impact: IBP

For IBP, we experiment with MNIST at $\epsilon_{\text{train}} = 0.4$ and $\epsilon_{\text{test}} = 0.3$; our results are in Table 6. Similar to PGD-AT, the small CNN has the greatest PGD and IBP verified robustness improvement with BN, at 2.7%; however, the medium and large CNN are roughly similar in their improvement with BN. Interestingly, CROWN-IBP does better than BN with the small CNN, presumably because CROWN-IBP provides a more direct mitigation of training instability with IBP (though computational cost is higher as well). The medium CNN with BN fared the best against CROWN-IBP in terms of clean accuracy and robustness. All BN-IBP models have significantly higher PGD accuracy compared to CROWN-IBP, potentially due to the train-test consistency that IBP's BN formulation provides.

	No BN (IBP)			No BN (CROWN-IBP)			With BN (IBP)		
Model	Clean	PGD	IBPV	Clean	PGD	IBPV	Clean	PGD	IBPV
SM-CNN	96.19	88.82	86.08	**97.04**	91.74	**90.56**	96.50	**92.96**	88.78
MD-CNN	97.55	92.18	90.60	97.63	92.26	91.46	**97.94**	**95.26**	**92.55**
LG-CNN	97.91	93.28	91.25	**98.18**	93.95	92.98	98.17	**96.11**	**93.38**

Table 6: IBP performance of various model sizes with/without BN on MNIST ($\epsilon_{\text{train}} = 0.4$, $\epsilon_{\text{test}} = 0.3$), with CROWN-IBP (no BN) for comparison. PGD column denotes accuracy under 200-step PGD attack. IBPV column denotes verified accuracy computed using IBP as the verification method.