Understanding the Energy vs. Adversarial Robustness Trade-Off in Deep Neural Networks

Kyungmi Lee, Anantha P. Chandrakasan
Department of Electrical Engineering & Computer Science
Massachusetts Institute of Technology
Cambridge, USA
kyungmi@mit.edu, anantha@mit.edu

Abstract—Adversarial examples, which are crafted by adding small inconspicuous perturbations to typical inputs in order to fool the prediction of a deep neural network (DNN), can pose a threat to security-critical applications, and robustness against adversarial examples is becoming an important factor for designing a DNN. In this work, we first examine the methodology for evaluating adversarial robustness that uses the first-order attack methods, and analyze three cases when this evaluation methodology overestimates robustness: 1) numerical saturation of cross-entropy loss, 2) non-differentiable functions in DNNs, and 3) ineffective initialization of the attack methods. For each case, we propose compensation methods that can be easily combined with the existing attack methods, thus provide a more precise evaluation of robustness.

Keywords—adversarial robustness, deep neural networks, energy-efficiency

I. INTRODUCTION

Deep Neural Networks (DNNs) are known to be vulnerable to adversarial examples, which are input samples that are intentionally perturbed to mislead the predictions of DNNs although they are often indistinguishable from typical “clean” input samples to human perception [1], [2] (Fig. 1). This vulnerability can be concerning when DNNs are to be deployed in high-stake applications, such as autonomous driving [3] or security cameras [4], where adversarial examples that fool the decision making can endanger the safety of those systems. Understanding robustness against adversarial examples when designing DNNs is important for such applications, especially in relation to other important design factors such as energy, and precise evaluation of robustness is an essential methodology to achieve this.

A practical methodology for evaluating robustness is using accuracy of DNNs on perturbed examples that are generated by the first-order attack methods as a proxy for robustness. Small perturbations that can change the prediction of DNNs are usually non-trivial (e.g., they are not random noises), and those attack methods are heuristic algorithms that propose perturbations using gradients of DNNs [5], [6]. While this approach is computationally efficient as the attack methods rely on back-propagation that can be easily accelerated by GPUs, the attack methods often fail to find effective perturbations that can change the prediction. For example, a well-studied phenomenon called gradient masking induces computation of gradients to be inaccurate [7], [8], and measured robustness of DNNs that are affected by gradient masking can be largely overestimated compared to their true robustness. Therefore, it is important to analyze when failures of such attack methods are not indicative of true robustness for more precise evaluation of robustness.

In this work, we identify three common cases in which failure of the first-order attack methods does not imply true robustness: 1) cross-entropy loss close to zero resulting in numerical saturation, 2) non-differentiable elements of DNNs, such as Rectified Linear Units (ReLU) and MaxPool, inducing "gradient shattering" [8], and 3) initialization for first-order searches being ineffective. We observe these phenomena are prevalent among conventionally trained DNNs across different architectures and datasets, not limited to specific defense methods that are known to cause gradient masking. For each case, we propose compensation methods that can be easily combined with the existing attack methods (Section III).

Next, we apply our proposed evaluation method to benchmark the relationship between adversarial robustness and energy required for DNN inference, which is an important design factor for resource constrained environments. In Section IV, we benchmark robustness of DNNs with different capacities,
pruning techniques, and regularization methods, and measure their inference-time energy on NVIDIA Jetson TX2. We demonstrate that overestimated robustness can obscure this relationship, and often result in a more pessimistic trade-off between them. Also, we show that the trade-off for robustness can be different from that for conventional clean accuracy.

II. PRELIMINARIES

A. Notations and definitions

We use $x$ to denote an unperturbed (clean) input and $f(\cdot)$ to represent a DNN. We only consider the classification task using cross-entropy loss. Given logits $z = f(x)$ and the ground truth label $t$, we express the loss as $l(z, t)$.

We consider adversarial robustness of DNNs when their model parameters are known to the attacker, so that gradients can be computed by the attacker using the parameters. We assume that the attacker adds small perturbations $r$ with the maximum size of $\epsilon$ (i.e., $\|r\|_p \leq \epsilon$, $p = 2, \infty$) to input samples $x$, to mislead the prediction of DNNs to any class other than the ground truth. Accuracy of a DNN against perturbed examples generated by such attack methods is a key metric for this work, and we refer this accuracy as attack accuracy, and distinguish it from clean accuracy which is a conventionally measured accuracy for clean input samples in the test set.

B. Related work

Adversarial examples Adversarial examples are recently investigated for DNNs, and many attack methods are proposed to efficiently generate perturbations [5, 6]. Notably, [5] found that first-order approximation of non-linear DNNs can be successful in finding adversarial examples, and [6] applied this first-order approximation iteratively to obtain stronger attack methods. For example, to generate a perturbation $r$ under $L_\infty$ norm (i.e., $\|r\|_\infty \leq \epsilon$), Projected Gradient Descent (PGD) [6] updates $r^{(i)}$ in the $i$th iteration as:

$$r^{(i)} = r^{(i-1)} + \alpha \cdot \text{sign}(\frac{\partial l(f(x + r^{(i-1)}), t)}{\partial r^{(i-1)}})$$

(1)

where initialization, $r^{(0)}$, is randomly sampled, and $\alpha$ represents a step size per iteration. Note that $r^{(i)}$ has to be clipped such that $\|r^{(i)}\|_\infty \leq \epsilon$ and $x + r^{(i)}$ is in the valid input domain. If the number of iterations is $1$ and $r^{(0)}$ is set to $0$, the above equation boils down to Fast Gradient Sign Method (FGSM) [5] that uses a single step of first-order approximation. Also, recent work from [9] replaces a hyperparameter $\alpha$ in (1) with an automatically optimized step size (AutoPGD), such that PGD becomes less sensitive to the choice of hyperparameters.

Gradient masking [7] observed that a defense method using knowledge distillation [10] shows high attack accuracy, but is still susceptible to perturbations generated from other DNNs. [7] noted that gradients of the defended DNN can be hidden from the attacker, leading to high attack accuracy, and called this phenomenon gradient masking. [8] also found that some defense methods relied on inaccurate computation of gradients, by adding stochastic or non-differentiable operations during inference.

Relationship between adversarial robustness and other design factors The trade-off between conventional clean accuracy and adversarial robustness has been investigated in [11] that showed this trade-off with a theoretical analysis on a linear supporting vector machine, and [12] that demonstrated this trade-off with empirical experiments on diverse DNN architectures. Furthermore, other studies showed how popular model compression techniques can be related to robustness [13].

C. Experimental setup

Dataset We use CIFAR-10 [14] to benchmark attack accuracy and energy, and additionally use SVHN [15] and TinyImageNet (a down-sampled dataset from [16]) to analyze and verify failure cases. The images are normalized to the range [0, 1] for both training and testing, and further pre-processing includes random crop and flips during training. We randomly split 10% of training samples for validation purpose.

DNN architectures We examine various design choices, including different model capacity, usage of Batch Normalization (BN) [17], and residual connections [18]. We consider a simple architecture with 4 convolution layers and 2 fully-connected layers (Conv4FC2), and a variant of it using BN (Conv4FC2-BN). Also, we examine popular architectures, such as WideResNet (WRN) [19] and MobileNetV2 [20].

Model training We train DNNs for 100 epochs using stochastic gradient descent with momentum as a default setup. We state regularization techniques, such as weight decay or adversarial training, when they are used for certain DNNs in Section IV. Otherwise, no explicit regularization technique is applied. Typically, the trained DNNs show clean accuracy of 80-95% for CIFAR-10, 95% for SVHN, and 40-60% (top-1) for TinyImageNet, depending on architectures.

Implementation All experiments are implemented with PyTorch [21]. We use AdverTorch [22] as a framework to implement attack methods. We investigate popular first-order attack methods, such as FGSM and PGD, along with their modified versions (e.g., AutoPGD).

Energy measurement platform For Section IV, we measure energy consumption for DNN inference using NVIDIA Jetson TX2 as a platform. We use the tegrastats utility to monitor power consumption of CPU, GPU, and DDR on this platform every 1 millisecond.

III. ANALYSIS ON FAILURE MODES OF ATTACKS

In this section, we analyze when the first-order attack methods fail to find adversarial perturbations, and identify three cases in which such failure does not indicate robustness (Fig. 2). We provide compensation methods for each case to improve the evaluation metric.

A. Zero loss

Cross-entropy loss gets small when pre-softmax logits have large "margin", the gap between logits corresponding to the most likely and the second most likely labels. Note that logit margins can be simply inflated by weights with large
magnitudes, for instance when no regularization that can penalize large weights is applied. Furthermore, exponential and logarithmic operations involved in computing cross-entropy loss can numerically saturate the computed value of loss when logit margins are large. To illustrate, consider $-\log \left( \frac{\exp(a) + \exp(b)}{\exp(b)} \right)$, which can be thought as cross-entropy loss for logits with two elements, $a$ and $b$. While mathematically this expression is larger than zero unless $b = -\infty$, we observe that it saturates to zero when $a - b \gtrsim 18$ using 32-bit floating-point representations in NumPy [23] and PyTorch [21] frameworks. When the value of loss becomes zero, the computed gradients naturally do not provide meaningful information, and resulting perturbations can be ineffective.

**Compensation** A straightforward way to account for this zero loss phenomenon is to ensure that the value of loss is sufficiently large so that gradients can be computed accurately. Once the value of loss is large, first-order attack methods can be applied as usual. To achieve this, we consider changing the target label when computing cross-entropy loss from the ground truth label $t$ to another class $t'$, such as the second most-likely class, which essentially gives the same expression as targeted attack methods that attempt to mislead a DNN’s prediction to a specific class. This approach can increase the value of loss since it computes cross-entropy loss with respect to the wrong label.

**B. Innate non-differentiability**

During back-propagation, gradients cannot be computed through non-differentiable functions in the computation graph. This property can be intentionally used to cause gradient masking; for example, a non-differentiable pre-processing step (e.g., bit-depth reduction [24]) ‘shatters’ gradient computation since the gradients with respect to the original input sample cannot be obtained due to the pre-processing step [8]. Here we show that innate non-differentiability of Rectified Linear Unit (ReLU) and MaxPool, which are popular choices in DNN architectures, also subtly affects gradient computation.

Gradients are not passed through negative-valued neurons using ReLU as an activation function, as they are set to zero by ReLU. However, when perturbations generated using these gradients are added to inputs and forward-propagated during evaluation, perturbations can “switch” some of previously negative-valued neurons to take positive values, and vice versa. MaxPool has a similar problem as ReLU, when added perturbations change the maximum-valued neuron in a given window to another neuron that was previously ignored when computing gradients. The “switching” of ReLU and MaxPool between forward- and backward-propagation is problematic for first-order attack methods, since local linear approximation is not possible when the effective neurons contributing to output logits change.

**Compensation** Backward Pass Differentiable Attack (BPDA) allows approximating gradients through a non-differentiable function $g(x)$ by substituting this function with a similar but differentiable function $h(x)$ during back-propagation [8]. We find that BPDA can be repurposed to compensate for the switching of ReLU and MaxPool. We substitute ReLU with other activation functions that are differentiable at zero but with similar behavior as ReLU, such as softplus function $h(x) = \frac{1}{2} \log(1 + \exp(\beta x))$. Also, MaxPool can be interpreted as $L_{\infty}$-norm pooling, and a natural differentiable substitute function will be $L_p$-norm pooling with sufficiently large $p$.

**C. Ineffective initialization**

Random initialization of PGD ($r^{(0)}$ in (1)) can affect whether the attack method will successfully find an adversarial perturbation [6]. We observe that it is more difficult for the models trained with certain conditions, such as excessive weight decay (i.e., increasing the strength of weight decay as training progresses), to find “good” initialization that result in successful perturbations within a few iterations, compared to other models. For example, a WRN model trained using excessive weight decay on CIFAR-10 is evaluated to have 7.11% of PGD attack accuracy ($\epsilon = 0.5$, $L_2$, compensated for zero loss and non-differentiability) compared to only 0.12% of a model with no regularization when PGD uses 5 iterations, but it ultimately shows less than 0.1% of PGD attack accuracy when the number of iterations is increased to 100. We further observe that this PGD attack with 5 iterations needs 68.2 restarts on average for 7.11% of input samples on which it fails. From these observations, we hypothesize that certain training conditions, such as the combination of the WRN architecture and excessive weight decay, can affect PGD initialization more difficult, thus increase the number of iterations to find adversarial perturbations.

**Compensation** Although using a large number of iterations is an uncomplicated way to prevent overestimation of attack accuracy when this phenomenon occurs, we investigate methods that can utilize the above hypothesis and reduce the number of iterations needed. The main idea is to set initialization to points where gradients increase rapidly, thus with high curvature in loss, so that subsequent first-order searches can find successful perturbations quickly. However, a challenge is that the exact curvature of loss requires the second-order information that is computationally expensive and numerically
unstable. Therefore, we adopt approximation techniques to find such initialization. Specifically, we consider approximating the principal eigenvector, which corresponds to the largest eigenvalue, of the Hessian matrix \( H = \nabla^2 \ell(f(x), t) \) using power iteration method and finite difference method proposed in [25], and use it as a direction for initialization. Computationally, this method requires two additional back-propagations to find initialization. In practice, when an iterative attack uses total \( N \) back-propagations, this initialization step will use 2 back-propagations and \( N - 2 \) first-order iterations will be applied subsequently.

IV. ENERGY-ROBUSTNESS TRADE-OFF

In this section, we demonstrate how the three phenomena that cause overestimation of attack accuracy affect the empirically benchmarked relationship between attack accuracy and energy consumption. This relationship can help designing DNNs for security-critical applications running at resource-constrained embedded devices.

A. Training models with different architectures and widths

1) Trade-off curves for attack accuracy and energy: We benchmark the relationship between attack accuracy and energy for DNNs with various architectures and widths (i.e., the number of neurons per layer). We train DNNs with Conv4FC2, Conv4FC2-BN, WRN, and MobileNetV2 architectures, and with 6 different relative width scaling factors on the CIFAR-10 dataset. We measure attack accuracy of these models with and without the compensation methods proposed in Section III, under the same computational budget (e.g., the number of back-propagations and restarts). Energy consumed for inference is measured on NVIDIA Jetson TX2.

Fig. 3 shows the relationship between attack accuracy and energy per a single input image when using PGD with \( \epsilon = 0.3 \) in \( L_2 \)-norm. First, note that the relationship between attack accuracy and energy can be significantly changed when the compensation methods are applied. For example, attack accuracy of a MobileNetV2 model with the largest width decreases by 34.9% when the compensation methods are applied. Consequently, MobileNetV2 models can appear to have good \( L_2 \)-norm PGD attack accuracy for \( 5 \sim 10 \)mJ energy budget without proper compensation, although they only have \( \sim 1\% \) attack accuracy in fact. Therefore, comparison of the trade-off curves for adversarial robustness has to be careful about the sources of overestimated attack accuracy.

Furthermore, the gap between attack accuracy before and after applying the compensation methods depends on the type of attack method and DNN architectures. Generally, we observe that DNNs with Conv4FC2 architecture, which does not have BN or residual connections, are least affected by the compensation methods across different attack methods. Also, \( L_\infty \)-norm PGD is least affected among attack methods studied here, and FGSM is more severely affected compared to iterative attack methods for both \( L_\infty \) and \( L_2 \)-norm.

Finally, we emphasize that the trade-off between attack accuracy and energy is different from the trade-off for conventional clean accuracy and energy. For example, Conv4FC2 models have the simplest architecture and the lowest clean accuracy compared to other DNNs (low energy, low clean accuracy), but their attack accuracy is higher than others (low energy, high attack accuracy). Similar to [11] and [12], we observe that DNN architectures designed for high clean accuracy, such as MobileNetV2 or WRN, show low attack accuracy, resulting in different trade-offs for energy. Therefore, DNN architectures that are well-known for high clean accuracy while being energy-efficient cannot be simply adopted for applications requiring robustness.

2) Influence of regularization techniques: We further analyze how regularization techniques affect the relationship between attack accuracy and energy for DNNs studied above. We examine popular regularization techniques that have been proposed for better generalization or robustness: weight decay, which adds \( L_2 \)-norm penalty on weight matrices; spectral normalization [26], which sets the largest singular value of each weight matrix to 1; orthonormal regularization [27], which penalizes non-orthonormal weight matrices so that all eigenvalues of weight matrices to be 1; input-gradient regularization [28], which penalizes gradients \( g \) to reduce the first-order term in Taylor’s expansion for \( l(f(x+r), t) \); adversarial training [5], [6], which trains a DNN on the perturbed input samples generated by attack methods, such as FGSM and PGD.

To concisely show the trade-off between attack accuracy and energy, we quantify the gain of attack accuracy per mJ of energy by comparing DNNs with different widths for each regularization technique (Table I). We observe that applying the compensation methods reduces the “accuracy/mJ” gain for some regularization techniques. For example, no explicit regularization (\( \Delta = -0.746 \)) and spectral normalization
(Δ = -1.173) are affected the most, possibly because they are more susceptible to the zero-loss phenomenon. For these cases, the trade-off for attack accuracy appears to be more larger than that for clean accuracy without the compensation methods, although they are similar. This example shows that overestimation can be more severe for certain regularization techniques, and comparison of these techniques has to consider the evaluation metric carefully.

B. Pruning

1) Weight pruning: As another approach to benchmark the trade-off between attack accuracy and energy, we investigate pruning, which is popularly used to reduce the number of weights in over-parameterized DNNs without losing clean accuracy. First, we consider weight pruning that removes weights with small magnitude without imposing explicit structure for sparsity [29]. We experiment with two large WRN models trained and pruned on CIFAR-10 under the same condition except for that one uses no explicit regularization and another model uses weight decay. We adopt iterative pruning that removes 25% of weights and finetunes for 10 epochs per iteration. Without explicit regularization, baseline PGD attack accuracy drops significantly (> 25%) as more weights are pruned away (Fig. 4(a)). However, applying the compensation methods shows that attack accuracy actually drops less than 1%, similar to that of clean accuracy. The major source of discrepancy is that the original dense model’s attack accuracy is overestimated due to the zero-loss phenomenon phenomenon.

On the other hand, baseline PGD attack accuracy increases by 3.5% at the end of pruning with weight decay (Fig 4(b)). However, when the compensation methods are applied, attack accuracy shows less than 0.4% of increase. We find that weight decay used during finetuning, which adds a large number of epochs (e.g., 10 epochs for finetuning × 10 pruning iteration → 100 additional epochs), can act similar to excessive weight decay discussed in Section III-C. As a result, sparser models show higher baseline attack accuracy, and using initialization methods of Section III-C gives a more accurate evaluation in this scenario.

![Fig. 4.](image)

This example illustrates how compensating for the phenomena discussed in Section III can prevent misleading conclusions. For example, for WRN models and $L_2$ PGD attacks we tested here, pruning does not affect attack accuracy more than 1%. However, without proper compensation methods, one might conclude that pruning negatively affects attack accuracy only observing the case without weight decay, or that pruning improves attack accuracy while also reducing the model size after experimenting with weight decay.

2) Channel pruning: To directly obtain the trade-off between attack accuracy and energy, we analyze $L_1$-norm channel pruning that keeps dense computation structure in the resulting sparse DNN [30]. Note that DNNs pruned with this method can be easily accelerated by commercial GPUs. Here, we analyze four different DNN architectures studied in Section IV-A1. We remove 10% of filters with the smallest $L_1$-norm, and finetune for 10 epochs per each pruning iteration without explicit regularization (also for the initial training), similar as in weight pruning.

Fig. 5 shows the relationship between the change in attack accuracy and energy as pruning progresses for each type of DNN architecture. For WRN and MobileNetV2, we observe that the trade-off can appear to be more pessimistic without the compensation methods. For example, baseline PGD attack accuracy drops by 16.22%, but applying the compensation methods

| Regularization   | Accuracy (%) / mJ | Δ         |
|------------------|------------------|-----------|
|                  | Clean | Baseline | Comp |          |
| None             | 0.176 | 0.828    | 0.082 | -0.746   |
| Weight Decay     | 0.162 | 0.267    | 0.194 | -0.073   |
| Spectral         | 0.236 | 1.534    | 0.362 | -1.173   |
| Orthonormal      | 0.171 | 0.191    | 0.185 | -0.006   |
| Input Gradient   | 0.248 | 0.000    | 0.026 | 0.026    |
| Adv. Training    | 0.342 | 0.311    | 0.316 | 0.005    |

The gain in PGD attack accuracy ($L_2$-norm, $\epsilon = 0.3, 9$ back-propagations, in %) per MJ of energy for WRN models trained with different regularization techniques. We measure the gain by comparing WRN models with the smallest and the largest widths in Fig. 3. $\Delta$ is the difference between the gain measured for compensated attack accuracy and baseline attack accuracy. The gain for clean accuracy is also presented as a reference.
reveals that the change is only 1.35% for MobileNetV2.

V. CONCLUSION

In this work, we demonstrate that sources of overestimated attack accuracy exist for many conventionally trained DNNs. We identify three sources of overestimation, and propose compensation methods for each of them. We further contribute by illustrating how these three sources affect benchmarking of the trade-off between robustness and energy. In particular, we show the comparison of different architectures for their energy-robustness trade-off, and the impact of pruning on attack accuracy. We think future work on developing energy-efficient DNNs for robustness can be beneficial to security-critical systems at embedded hardware platforms.

ACKNOWLEDGMENT

This work was sponsored by MIT Jacobs Presidential Fellowship, Siebel Scholars Foundation, Korea Foundation for Advanced Studies, and NXP Semiconductors.

REFERENCES

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” 2013.
[2] B. Biggio and F. Roli, “Wild patterns: Ten years after the rise of adversarial machine learning,” Pattern Recognition, vol. 84, p. 317–331, Dec 2018. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2018.07.023
[3] K. Eykholt, I.Evtimov, E. Fernandez, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, “Robust physical-world attacks on deep learning models,” 2017.
[4] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, “Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition,” in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ser. CCS ’16. New York, NY, USA: ACM, 2016, pp. 1528–1540. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978392
[5] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” arXiv preprint arXiv:1412.6572, 2014.
[6] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” arXiv preprint arXiv:1706.06083, 2017.
[7] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, “Practical black-box attacks against machine learning,” in Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, ser. ASIA CCS ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 506–519. [Online]. Available: https://doi.org/10.1145/3052973.3053009
[8] A. Athalye, N. Carlini, and D. Wagner, “Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples,” 2018.
[9] F. Croce and M. Hein, “Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks,” 2020.
[10] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, “Distillation as a defense to adversarial perturbations against deep neural networks,” 2016 IEEE Symposium on Security and Privacy (SP), May 2016. [Online]. Available: http://dx.doi.org/10.1109/sp.2016.41
[11] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, “Robustness may be at odds with accuracy - a comprehensive study on the robustness of 18 deep image classification models,” ArXiv, vol. abs/1808.01688, 2018.
[12] J. Lin, C. Gan, and S. Han, “Defensive quantization: When efficiency meets robustness,” in Advances in Neural Information Processing Systems, 2017.
[13] A. Prakash, T. Kohno, and D. Song, “Robust physical-world attacks on deep learning models resistant to adversarial examples,” in Conference on Learning Representations, 2019. [Online]. Available: https://openreview.net/forum?id=rJ7Zc8cX
[14] A. Krizhevsky, G. Hinton et al., “Learning multiple layers of features from tiny images,” Citeseer, Tech. Rep., 2009.
[15] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, “Reading digits in natural images with unsupervised feature learning,” 2011.
[16] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” International Journal of Computer Vision (IJCV), vol. 115, no. 3, pp. 211–252, 2015.
[17] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” 2015.
[18] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jun 2016. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2016.90
[19] S. Zagoruyko and N. Komodakis, “Wide residual networks,” 2016.
[20] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetv2: Inverted residuals and linear bottlenecks,” 2019.
[21] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in PyTorch,” in NIPS Autodiff Workshop, 2017.
[22] G. W. Ding, L. Wang, and X. Jin, “AdvTorCh v0.1: An adversarial robustness toolbox based on pytorch,” arXiv preprint arXiv:1902.07623, 2019.
[23] T. E. Oliphant, A guide to NumPy. Trelgol Publishing USA, 2006, vol. 1.
[24] C. Guo, M. Rana, M. Cisse, and L. van der Maaten, “Countering adversarial images using input transformations,” in International Conference on Learning Representations, 2018. [Online]. Available: https://openreview.net/forum?id=SycJCIWcb
[25] T. Miyato, S.-I. Maeda, M. Koyama, and Y. Ishii, “Virtual adversarial training: A regularization method for supervised and semi-supervised learning,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, no. 8, pp. 1979–1993, Aug 2019. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2018.2858821
[26] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, “Spectral normalization for generative adversarial networks,” in International Conference on Learning Representations, 2018. [Online]. Available: https://openreview.net/forum?id=B1QRgztT-
[27] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier, “ParSearch networks: Improving robustness to adversarial examples,” 2017.
[28] A. S. Ross and F. Doshi-Velez, “Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients,” 2017.
[29] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” 2015.
[30] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, “Pruning filters for efficient convnets,” 2017.