3rd Symposium on Advances in Approximate Bayesian Inference, 2020 1–11

Efficient Calculation of Adversarial Examples for Bayesian Neural Networks

Anonymous Authors
Anonymous Institution

Abstract

Calculating adversarial examples for Bayesian neural networks is cumbersome. Due to the inherent stochasticity, the gradient of the network can only be reliable approximated by sampling multiple times from the posterior, leading to a greatly increased computational cost. In this paper we propose to efficiently attack Bayesian neural networks with adversarial examples calculated for a deterministic network with parameters given by the mean of the posterior distribution. We show in a series of experiments, that the proposed approach can be used to effectively attack Bayesian neural networks while using 4.2 times less of the resources of existing adversarial example estimation methods with comparable strength. We demonstrate that this is especially helpful during adversarial training, when multiple different model configuration need to be evaluated.

1. Introduction

Machine learning is ubiquitous. Starting from breakthroughs in computer vision (Krizhevsky et al., 2012; He et al., 2016) and natural language processing (Collobert and Weston, 2008); machine learning systems have evolved to play games at human level (Silver et al., 2016; Vinyals et al., 2019; Mnih et al., 2015), leverage drug discovery (Senior et al., 2020), recognize speech at the level of human listeners (Ram et al., 2017), and drive cars (Shankland, 2019). Given the influence on more and more parts of human live, the existence of adversarial examples (AEs) (Biggio et al., 2014; Szegedy et al., 2014) is worrisome. These small perturbations, added to benign inputs, cause even state-of-the-art models to misbehave.

Recent work suggest Bayesian neural networks (BNNs) (Neal, 1995) as a more robust alternative (Feinman et al., 2017; Gal and Smith, 2018; Liu et al., 2019; Bekasov and Murray, 2018). Carbone et al. (2020) even show that under certain assumptions the gradient has zero expectation under the posterior distribution and hence any gradient based attack in the limit is ineffective. Despite these theoretical efforts BNNs have successfully been attacked by taking multiple samples from the posterior distribution to approximate their gradient as proposed as a general attack strategy for networks with stochastic gradients (Athalye et al., 2018; Zimmermann, 2019). While effective, this approach is computationally costly. This is less of a problem for a malicious actor, since they simply need to find one AE to serve their purpose. Yet, from the defending parties point of view, evaluating several different models with regards to adversarial robustness can quickly become a costly endeavor. This cost is severely multiplied in the context of adversarial training (Madry et al., 2017), where adversarial examples are used in the training process to generate robust models.

Recognizing these problems, we introduce `MEAN`—an efficient way to compute AEs for BNNs, which can easily be adopted to other types of stochastic networks. Our approach

© A. Authors.

employs the means of the weight distribution as the parameters of a deterministic network of equal architecture. We then employ this network to estimate AEs which are then used to attack the original BNN. In a first experiment, we demonstrate that `MEAN` can be used to generate effective AEs while using substantially fewer resources than existing approaches. In a second experiment, we evaluate `MEAN` in the setting of adversarial training, where a computational efficient method allows a defending party to evaluate multiple different models with a much smaller computational budget.

2. Background and Approach

Notation: We denote with $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1}^n\}$ the training data where $\boldsymbol{x}_i$ are inputs, $\boldsymbol{y}_i$ the corresponding outputs and $\boldsymbol{\theta}$ the network's parameters.

2.1. Background

Bayesian neural networks serve as a mathematical quantified way to capture model uncertainty by marginalizing over the posterior distribution of the network's parameters:

$$p(\boldsymbol{y}|\boldsymbol{x}, \mathcal{D}) = \int p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta} \approx \frac{1}{S} \sum_{s=1}^{S} p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}_s) \ , \tag{1}$$

where the last approximation is the Monte Carlo approximation of this intractable integral. Central to the training of BNNs is the derivation of the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$. In this work we use variational inference (Graves, 2011; Kingma et al., 2015; Louizos and Welling, 2016) where the true posterior is approximated via a simpler distribution $q(\boldsymbol{\theta})$ by maximizing the so called *evidence lower bound* (ELBO):

$$ELBO = \sum_{i=1}^{n} \mathbb{E}_{q(\boldsymbol{\theta})}[p(\boldsymbol{y}_i|\boldsymbol{x}_i, \boldsymbol{\theta})] - KL[q(\boldsymbol{\theta})||p(\boldsymbol{\theta})] \ , \tag{2}$$

where the last term is the Kullback-Leibler divergence between the approximate posterior and some prior distribution.

Adversarial examples are manipulated inputs aiming at changing the networks' behavior. They are created by adding a well-calibrated perturbation δ to a benign input. One of the simplest methods to determine this δ is the *Fast Gradient Sign Method* (FGSM) (Goodfellow et al., 2015), in which the AE is created by adding an ϵ-step into the direction of the gradient sign of some loss function L, e.g.:

$$\boldsymbol{x}_{adv} = \boldsymbol{x} + \epsilon \cdot sign\left(\nabla_{\boldsymbol{x}} L\left[\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y}\right]\right) \ . \tag{3}$$

In practice the δ is usually limited by an L_p-norm to avoid changing the benign input too much. Improving FSGM, *Projected Gradient Decent* (PGD) (Madry et al., 2017) is an iterative attack which iterativly applies FGSM with a smaller stepsize than ϵ.

To calculate AE for BNNs, multiple draws from the posterior are needed for reliably approximating the stochastic gradient. For the FGSM attack this results in:

$$\boldsymbol{x}_{adv,BNN} = \boldsymbol{x} + \epsilon \cdot sign\left(\sum_{t=1}^{T} \nabla_{\boldsymbol{x}} L\left[\boldsymbol{\theta}_t, \boldsymbol{x}, \boldsymbol{y}\right]\right) \ . \tag{4}$$

In this work, we use the loss function introduced by Carlini and Wagner (2017) instead of the more commonly used cross-entropy loss. The loss function is given by

$$L_{\text{CW}}(\boldsymbol{x}, \boldsymbol{\theta}) = \max(\max_{i \neq t}(Z(\boldsymbol{x}, \boldsymbol{\theta})_i) - Z(\boldsymbol{x}, \boldsymbol{\theta})_t, \mathbf{0}) \ , \tag{5}$$

where $Z(\boldsymbol{x}, \boldsymbol{\theta})_t$ is an arbitrary logit of an output node for input $\boldsymbol{x}$ and parameters $\boldsymbol{\theta}$. We noticed that our networks often saturated the softmax function, i.e., the networks were so confident in their prediction that the probability of the correct class effectively reached 1. This resulted in us being unable to calculate gradients. However, since the Carlini-Wagner loss function operates on the logits of the classifier, it enables the creation of adversarial examples.

2.2. Approach

Our approach aims at reducing the cost of sampling from the posterior while at the same time generating a strong attack. This is done by transforming the stochastic gradients in eq. (4) to deterministic ones by treating the BNN as a standard neural network where the posterior means serve as the weights value. We refer to this scheme of replacing the stochastic neural network with a deterministic one as `MEAN`. Note that this approach does not restrict the set of possible adversarial attacks. Under `MEAN` eq. (4) reduces to:

$$\boldsymbol{x}_{adv,\texttt{MEAN}} = \boldsymbol{x} + \epsilon \cdot sign\left(\nabla_{\boldsymbol{x}} L\left[\mathbb{E}[\boldsymbol{\theta}], \boldsymbol{x}, \boldsymbol{y}\right]\right) \ . \tag{6}$$

Note the similarity of our approach to the rescaling of weights in dropout networks (Srivastava et al., 2014), where weights are scaled to have the same expectation under the droprate during testing as during training.

3. Experiments

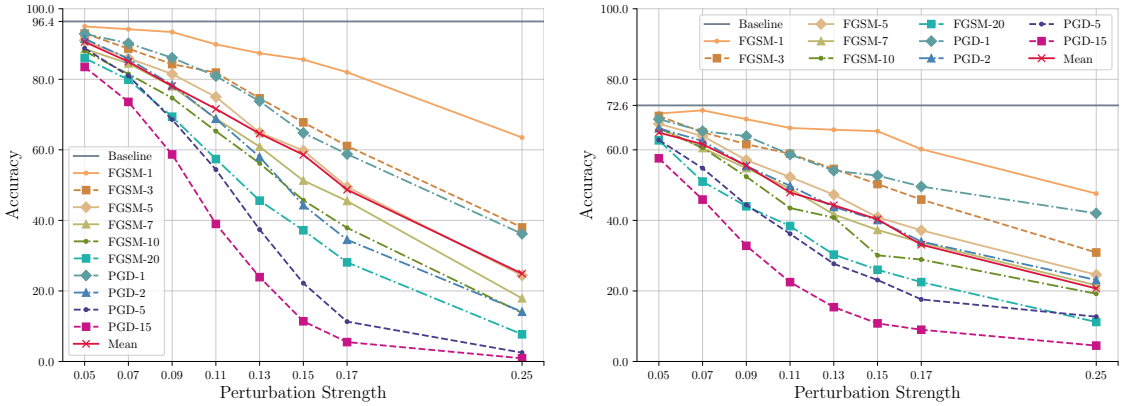
In this section we evaluate the benefits of the `MEAN` scheme for creating AEs, in two settings: as an attack and during adversarial training. We first describe the general experimental setting in Section 3.1 and compare the effectiveness of adversarial attacks conducted with `MEAN` in Section 3.2. Finally, in Section 3.3 we leverage the `MEAN` approach in the context of adversarial training.

3.1. Experiment Setup

In our experiments we use the MNIST and Fashion MNIST data sets (LeCun et al., 1998; Xiao et al., 2017), both consisting of 60,000 training and 10,000 test images of size 28×28 labeled as one out of 10 classes. We trained two different CNN networks[1] using mean field variational inference and Adam (Kingma et al., 2015) as the optimizer. All our experiments are conducted on a Nvidia Quadro RTX 5000 graphic card.

AEs are generated using the FGSM and PGD (with 10 iteration steps) methods with varying perturbation strengths. Note that during the evaluation we refer to the FGSM attack computed over 10 samples drawn from the posterior as FGSM-10 (15 samples—FGSM-15; and so forth).

1. Details on the architecture can be found in Section A in the Appendix.



(a) MNIST (b) Fashion MNIST

Figure 1: **Accuracy of adversarial examples generated using different sample sizes from the posterior and the `MEAN` scheme.** We report the accuracy depending on the perturbation strength used (δ) for both MNIST and Fashion MNIST. For generating the AEs we used FGSM and PGD attack. We used FGSM exclusively for the `MEAN` scheme. For reference we include the original accuracy on the test set (Baseline). Best viewed in color.

3.2. Generating Adversarial Examples

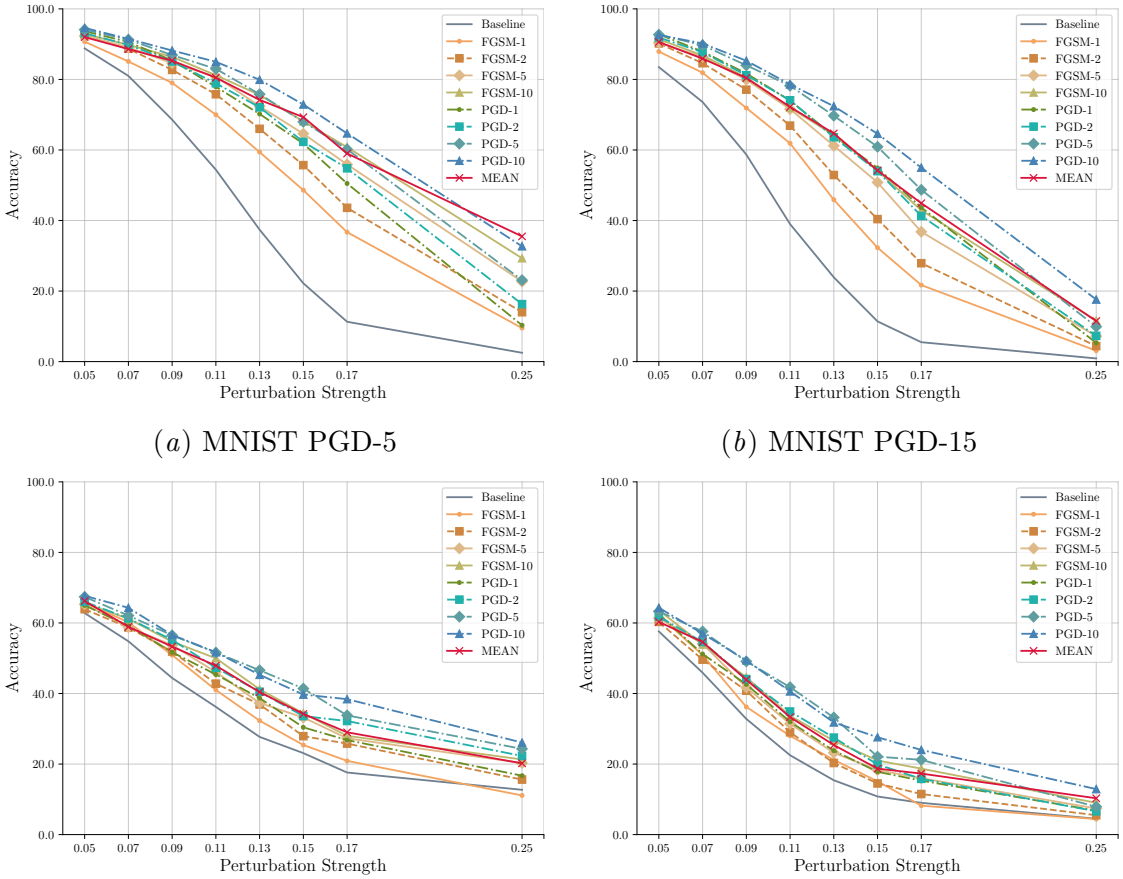
In this section we want to evaluate whether AEs generated by the `MEAN` scheme are comparable to examples generated by sampling from the posterior. Additionally, we will also evaluate the computational cost associated with the different approaches.

We generate 1,000 AE for $\epsilon \in \{0.05, 0.07, 0.09, 0.11, 0.13, 0.17, 0.25\}$ and report the mean accuracy. The results are depict in Figure 1. When compared to the FGSM attack, `MEAN` is competitive up to sampling five (MNIST) or seven (Fashion MNIST) times from the posterior. We independently measured how long it takes to convert a batch of images to AEs, by converting 10,000 batches and computing the average time. The full results can be found in Table 2 in Appendix B. For MNIST, `MEAN` takes about 88ms to convert a single batch while FGSM-5 takes about 366ms, an increase in 315.91 %. The results are mirrored for Fashion MNIST, where FGSM-7 requires around 554 % more time; proving that `MEAN` can be used to approximate computationally more expensive attacks.

When compared to the PGD attack, `MEAN` is stronger than PGD-1. It is on par with PGD-2 on Fashion MNIST, but otherwise outperformed. While surprising at first, we hypothesize that PGD, by its iterative nature, approximates the stochastic gradient quite well. Since PGD is essentially iterated FGSM, at each iteration we draw a new sample from the posterior. Thus, even PGD-1 can intuitively be thought of as sampling 10 times from the posterior (for 10 iteration steps).

Note, we exclusively used FGSM to generate the AEs with the `MEAN` scheme. During our experiments we discovered that using PGD in conjunction with the deterministic network created by the `MEAN` scheme would overfit to this specific network, resulting in worse overall performance.

3.3. Adversarial Training



(a) MNIST PGD-5

(b) MNIST PGD-15

(c) Fashion MNIST PGD-5

(d) Fashion MNIST PGD-15

Figure 2: **Robustness comparison of differently adversarial trained models against PGD attacks.** All adversarial trained networks were attacked with the PGD attack for $\{5, 15\}$ draws from the posterior. A higher accuracy stands for a higher robustness against adversarial attacks. We include the accuracy of a normal trained network for reference (Baseline). Best viewed in color.

As demonstrated in Section 3.2 the mean attack leads to significantly stronger adversarial examples than computational equivalent attacks. While the resource overhead of drawing multiple samples might be condoned by an attacker, this practice would be infeasible if a defending party wants to compare multiple models. Thus, we evaluate the effectiveness of `MEAN` in the context of adversarial training.

Prior work suggest using the strongest attack possible results in more robust models (Madry et al., 2017). However, in practice, often multiple model architecture and hyperparameter combinations have to be evaluated on different data sets. Thus, sampling multiple times quickly becomes infeasible. Additionally, robustness often requires a trade-off in regards to accuracy (Tsipras et al., 2019), leading to evaluating even more combinations

Table 1: **Training time for different adversarial training techniques.** We report the time it took to train a model in seconds and the increase relative to the `MEAN` in brackets.

Dataset	`MEAN`	FGSM-10		PGD-2		PGD-5		PGD-10	
MNIST	134s	208s	(+55 %)	304s	(+126 %)	592s	(+341 %)	1067s	(+696 %)
Fashion MNIST	336s	624s	(+85 %)	958s	(+185 %)	1871s	(+456 %)	3549s	(+956 %)

in practice. In this scenario `MEAN` can serve as a good approximation of stronger training techniques. It can be used in the early stages of development to gauge the resistance of networks to adversary examples, while the final model can be trained with the strongest attack possible.

We train multiple networks using different techniques to get a thorough understanding of how well we can utilize `MEAN` for training adversarial robust networks. We calculate adversarial examples on-the-fly during the training process, mixing "normal" and adversarial samples, i.e., with probability 0.3 we convert the current training batch to a batch of adversarial examples. The results can be found in Figure 2.

We plot the accuracy of different training schemes when attacked with PGD in different sample configurations ($\{5, 15\}$). Additionally, we provide the time it took to train the strongest models in Table 1 (a full table can be found in Appendix C). We can observe that `MEAN` can reliably approximate more costly techniques. It is only consistently outperformed by PGD-5 and PGD-10. Which require up to 956 % of the resources. Note that we only use adversarial examples in 30 % of the training, if we would supply the adversarial examples more often the discrepancy would be even higher.

4. Conclusion

Stochastic neural networks, like the here examined Bayesian neural networks (BNNs), incorporate an inherent gradient masking effect during the calculation of adversarial examples. Circumventing this masking comes at the cost of drawing multiple samples from the posterior for calculating the gradient. In this paper we proposed a simple and efficient solution to avoid the computational burden for BNNs, called `MEAN`. In this scheme, we transfer the BNN into a deterministic network, of equal architecture, by using the posterior means as weight values. In our experiments we found that using the `MEAN` scheme results in attacks as efficient as those produced by taking multiple samples from the gradient. Further, we found that `MEAN` has a high impact during adversarial training where is is only outperformed by PGD-5 and PGD-10, which require up to 956 % of the resources. Our findings suggest, that `MEAN` serves as a simple and cheap baseline for calculating adversarial examples. Therefore, `MEAN` is especially relevant during hyperparamerter tuning, extensive model comparison or costly network evaluations.

References

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

Artur Bekasov and Iain Murray. Bayesian Adversarial Spheres: Bayesian Inference and Adversarial Examples in a Noiseless Setting. In *Advances in Neural Information Processing Systems (NeurIPS) Bayesian Deep Learning Workshop*, 2018.

Battista Biggio, Giorgio Fumera, and Fabio Roli. Security Evaluation of Pattern Classifiers under Attack. *IEEE Transactions on Knowledge and Data Engineering*, 2014.

Ginevra Carbone, Matthew Wicker, Luca Laurenti, Andrea Patane, Luca Bortolussi, and Guido Sanguinetti. Robustness of Bayesian Neural Networks to Gradient-Based Attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (S&P)*, 2017.

Ronan Collobert and Jason Weston. A Inified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *International Conference on Machine Learning (ICML)*, 2008.

Reuben Feinman, Ryan R. Curtin, Saurabh Shintre, and Andrew B. Gardner. Detecting Adversarial Samples from Artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

Yarin Gal and Lewis Smith. Sufficient Conditions for Idealised Models to Have No Adversarial Examples: a Theoretical and Empirical Study with Bayesian Neural Networks. *arXiv preprint arXiv:1806.00667*, 2018.

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

Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2011.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Diederik P. Kingma, Tim Salimans, and Max Welling. Variational Dropout and the Local Reparameterization Trick. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 1998.

Xuanqing Liu, Yao Li, Chongruo Wu, and Cho-Jui Hsieh. Adv-BNN: Improved Adversarial Defense through Robust Bayesian Neural Network. In *International Conference on Learning Representations (ICLR)*, 2019.

Christos Louizos and Max Welling. Structured and Efficient Variational Deep Learning with Matrix Gaussian Posteriors. In *International Conference on Machine Learning (ICML)*, 2016.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Eesistant to Adversarial Attacks. In *International Conference on Learning Representations (ICLR)*, 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, et al. Human-Level Control through Deep Reinforcement Learning. *Nature*, 2015.

Radford M. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1995.

Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, et al. Conversational AI: The Science behind the Alexa Prize. *Advances in Neural Information Processing Systems (NeurIPS) Conversational AI Workshop*, 2017.

Andrew W. Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander W.R. Nelson, Alex Bridgland, et al. Improved Protein Structure Prediction using Potentials from Deep Learning. *Nature*, 2020.

Stephen Shankland. Meet Tesla's self-driving car computer and its two AI brains. https://www.cnet.com/news/ meet-tesla-self-driving-car-computer-and-its-two-ai-brains/, 2019.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 2016.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research (JMLR)*, 2014.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing Properties of Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2014.

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

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster Level in StarCraft II using Multi-Agent Reinforcement Learning. *Nature*, 2019.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical Evaluation of Rectified Activations in Convolutional Network. *arXiv preprint arXiv:1505.00853*, 2015.

Roland S. Zimmermann. Comment on "Adv-BNN: Improved Adversarial Defense through Robust Bayesian Neural Network". *arXiv preprint arXiv:1907.00895*, 2019.

Appendix

For the sake of completeness we extend this work by describing the implementation and adding further details on computational time for the results depict in the main part.

A. Network Architecture

The networks are constructed using pyro 1.3 and are based on the work of Carbone et al. (2020). For training the models we used mean field variational inference with standard Gaussian distributions as priors. We used ADAM (Kingma and Ba, 2015) as our optimizer with an initial learning rate of 0.01 (MNIST) and 0.001(Fashion MNIST). Further, we trained for 5 (MNIST) and 10 (Fashion MNIST) epochs with a batch size of 128 for both data sets. As our activation function we used the Leaky ReLu function (Xu et al., 2015) with a negative slope of 0.01 throughout the network.

MNIST
Conv 5x5 (32)
Max-Pooling (2)
Conv 5x5 (512)
Max-Pooling (1)
Dense (10)

Fashion MNIST
Conv 5x5 (32)
Max-Pooling (2)
Conv 5x5 (1024)
Max-Pooling (1)
Dense (10)

The network architecture for our simple CNN. We report in brackets: For convolution layers the amounts of feature maps learned, for dense layers the amount of hidden units, and, for pooling layers the amount of stride. We use exclusively max-pooling with a kernel size of two.

B. Time Required to Create Adversarial Examples

In this Section we report the time it took for converting AEs with different techniques. We convert a batch of images 10,000 times and average over all runs.

Table 2: **Time for converting AEs.** We report the time it takes to convert a single batch with the given technique, averaged over 10,000 samples. Additionally, we report the increase relative to the `MEAN` scheme in brackets.

Technique	MNIST		Fashion MNIST	
`MEAN`	88 ms		107 ms	
FGSM-1	122 ms	(+38.63 %)	119 ms	(+11.21 %)
FGSM-3	295 ms	(+235.22 %)	307 ms	(+186.92 %)
FGSM-5	366 ms	(+315.91 %)	504 ms	(+371.03 %)
FGSM-7	499 ms	(+467.05 %)	700 ms	(+554.21 %)
FGSM-10	614 ms	(+597.72 %)	996 ms	(+830.84 %)
FGSM-20	1,128 ms	(+1,181.81 %)	1,981 ms	(+1,751.40 %)
PGD-1	699 ms	(+ 694.32 %)	1,068 ms	(+898.13 %)
PGD-2	1,521 ms	(+ 1,628.41 %)	2,100 ms	(+1,862.62 %)
PGD-5	3,184 ms	(+ 3,518.18 %)	5,016 ms	(+4,587.85 %)
PGD-15	8,507 ms	(+ 9,567.05 %)	14,771 ms	(+13,704.67 %)

C. Training Time for Adversarial Training

In this Section we report the time it took to adversarialy train a network given the technique used to create AEs.

Table 3: **Training time for different adversarial training techniques.** We report the time it took to train a model in seconds and the increase relative to the `MEAN` in brackets.

Technique	MNIST		Fashion MNIST	
`MEAN`	134 s		336 s	
FGSM-1	133 s	(-0.87 %)	341 s	(+1.48 %)
FGSM-2	144 s	(+7.46 %)	372 s	(+10.71 %)
FGSM-5	169 s	(+26.12 %)	468 s	(+39.29 %)
FGSM-10	208 s	(+55.23 %)	642 s	(+91.07 %)
PGD-1	220 s	(+64.18 %)	650 s	(+93.45 %)
PGD-2	304 s	(+126.87 %)	958 s	(+185.12 %)
PGD-5	592 s	(+341.79 %)	1,871 s	(+456.85 %)
PGD-10	1,067 s	(+696.27 %)	3,549 s	(+956.25 %)