3rd Symposium on Advances in Approximate Bayesian Inference, 2021 1–12

Gradient-Free Adversarial Attacks for Bayesian Neural Networks

Matthew Yuan *Princeton University* MY4@PRINCETON.EDU
Matthew Wicker *University of Oxford* MATTHEW.WICKER@CS.OX.AC.UK
Luca Laurenti *University of Oxford* LUCA.LAURENTI@CS.OX.AC.UK

Abstract

The existence of adversarial examples, i.e., small perturbations to their inputs that can cause a misclassification, underscores the importance of understanding the robustness of machine learning models. Bayesian neural networks (BNNs), due to their calibrated uncertainty, have been shown to posses favorable adversarial robustness properties. However, when approximate Bayesian inference methods are employed, the adversarial robustness of BNNs is still not well understood. In this work, we employ gradient-free optimization methods in order to find adversarial examples for BNNs. In particular, we consider genetic algorithms, surrogate models, as well as zeroth order optimization methods and adapt them to the goal of finding adversarial examples for BNNs. In an empirical evaluation on the MNIST and Fashion MNIST datasets, we show that for various approximate Bayesian inference methods the usage of gradient-free algorithms can greatly improve the rate of finding adversarial examples compared to state-of-the-art gradient-based methods.

1. Introduction

Deep Neural Networks (NN) have shown state-of-the-art performance on many tasks, including image recognition (Krizhevsky et al., 2017) and reinforcement learning (Schulman et al., 2015). However, the vulnerability of NNs to adversarial examples questions their applicability to safety-critical scenarios, where a failure of a learning model can have catastrophic consequences (Goodfellow et al., 2014). Because of their principled treatment of uncertainty, Bayesian Neural Networks (BNNs), i.e., neural networks with a prior distribution placed over their parameters, have been proposed as a more robust learning model compared to standard SGD-trained NNs (Carbone et al., 2020; Bekasov and Murray, 2018). Nevertheless, the theoretical results are all limited to the idealised case where the posterior distribution of a BNN is computed exactly via the Bayes' rule. In practice, exact computation of the posterior is infeasible and approximate inference methods with a finite amount of data are employed (Neal, 2012). As a result, it is still unclear how to best find adversarial examples for commonly employed approximate Bayesian inference methods.

In this paper, we start from the observation that for fully trained BNNs the gradient of the loss tends to be uninformative (Carbone et al., 2020), and we adapt gradient-free optimization methods for finding adversarial attacks on BNNs. In particular, we consider: zeroth order optimization (Chen et al., 2017), optimization with surrogate gradients (Athalye et al., 2018), and genetic algorithms (Alzantot et al., 2019). We perform an empirical evaluation of various different NN architectures on both the MNIST and FashionMNIST datasets for networks trained with five different commonly employed approximate inference

© M. Yuan, M. Wicker & L. Laurenti.

methods for BNNs. We find that the presented algorithms can offer significant improvements, finding adversarial examples for up to 40% of images which remained correctly classified when using state-of-the-art, first-order attacks. [1]

2. Adversarial Attacks for BNNs

Bayesian modelling aims to capture the intrinsic uncertainty of data driven models. Consider a classification problem with n_C classes for a neural network $f^{\mathbf{w}}(x)$ with input $x \in \mathcal{R}^m$ and network parameters (weights and biases) $\mathbf{w}$, then one starts with a prior distribution over the network parameters $p(\mathbf{w})$. The fit of the weights $\mathbf{w}$ to the data D is computed through the likelihood $p(D|\mathbf{w})$. Bayesian inference combines likelihood and prior via the Bayes theorem to obtain a *posterior* distribution $p(\mathbf{w}|D) \propto p(D|\mathbf{w})\, p(\mathbf{w})$.

Computing the posterior distribution $p(\mathbf{w}|D)$ exactly is not possible in general for NNs. Asymptotically exact samples from $p(\mathbf{w}|D)$ can be obtained via approximate inference methods such as Hamiltonian Monte Carlo (HMC) (Neal et al., 2011), while approximate samples can be obtained more cheaply via Variational Inference (VI) (Blundell et al., 2015). Independently of the methods employed to compute the posterior distribution, predictions at a new input x^* are obtained from an ensemble of n NNs, each with its individual weights drawn from the approximate posterior distribution $p(\mathbf{w}|D)$:

$$\langle f^{\mathbf{w}}(x^*)\rangle_{p(\mathbf{w}|D)} \simeq \frac{1}{n}\sum_{i=1}^{n} f^{\mathbf{w}_i}(x^*) \qquad \mathbf{w}_i \sim p(\mathbf{w}|D) \tag{1}$$

where $\langle\cdot\rangle_p$ denotes expectation w.r.t. the distribution p. Eqn (1) is the expectation of the *predictive distribution* of the BNN.

Given an input point x^* and a strength (i.e. maximum perturbation magnitude) $\epsilon > 0$, an adversarial example is a point $\tilde{x}$ such that $|x^* - \tilde{x}| \leq \epsilon$ and

$$\text{argmax}_{i\in\{1,\ldots,n_C\}}\langle f_i^{\mathbf{w}}(x^*)\rangle_{p(\mathbf{w}|D)} \neq \text{argmax}_{i\in\{1,\ldots,n_C\}}\langle f_i^{\mathbf{w}}(\tilde{x})\rangle_{p(\mathbf{w}|D)},$$

where $f_i^{\mathbf{w}}$ is the i-th component of $f^{\mathbf{w}}$, and $|\cdot|$ is a suitable similarity metric (taken here to be the l_∞ norm). As $f^{\mathbf{w}}$ is non-linear, solving the above optimization problem exactly is infeasible and several approximate solution methods have been proposed. Among them gradient-based attacks are arguably the most prominent, efficient, and effective. In particular, these methods, which include FGSM (Goodfellow et al., 2014) and PGD (Madry et al., 2018), which rely on the gradient of the loss $L(x, \mathbf{w}_i)$ function wrt the input point x. However, it has been recently shown in (Carbone et al., 2020) that for overpametrised BNNs in the limit of high accuracy, exact inference, and infinite amount of data it holds that:

$$\langle \nabla_{\mathbf{x}} L(\mathbf{x}, \mathbf{w})\rangle_{p(\mathbf{w}|D)} = 0. \tag{2}$$

Such a vanishing behavior on the gradient (which is known to be not true for SGD-trained NNs) questions the efficacy of gradient-based attacks for BNNs. In what follows, we employ several optimization methods which do not directly rely on the gradient of the loss to attack BNNs and empirically evaluate them in Section 3.

1. All source code to reproduce these results including the posteriors used and BNN training scripts can be found at: https://github.com/matthewwicker/GradientFreeAttacksBNNs

2.1. Attacking BNNs without First-Order Information

Attacking deterministic neural networks when the gradient information is missing, unavailable, or uncomputable has been studied in prior works, see e.g., (Papernot et al., 2017; Athalye et al., 2018; Alzantot et al., 2019). Whereas several works have studied the robustness of BNNs (Cardelli et al., 2019b; Wicker et al., 2020; Michelmore et al., 2020) and developed more robust training methods (Liu et al., 2018), attacking a BNN when the gradient is uninformative or unavailable has not been considered in detail yet. Below, we adapt some of the more commonly gradient-free optimization approaches for this setting.

2.1.1. Zeroth Order Adversarial Attacks

As a first approach we consider numerical approximations of the gradient via Zeroth order optimization (ZOO) methods (Chen et al., 2017; Ilyas et al., 2018; Wicker et al., 2018). We consider these methods because it has been shown that for NNs with obfuscated gradients (i.e. when a NN is purposefully made non-differentiable or their gradients vanish as for BNNs) numerical approximations of the gradient can still offer useful information for finding adversarial attacks (Athalye et al., 2018; Papernot et al., 2017). In particular, these methods can be adapted to BNNs by making various queries to the predictive distribution to compute a finite difference approximation of the gradient of the expectation of the loss. The resulting approximated gradient is then feed into standard gradient-based algorithms, such as the fast gradient sign method (FGSM) (Goodfellow et al., 2014) and projected gradient descent (PGD) (Madry et al., 2018).

2.1.2. Backwards Pass Differentiable Approximations (BPDA)

Backwards Pass Differentiable Approximation (BPDA) attacks aim to learn a differentiable surrogate model in order to mimic the decision boundary of the classifier under consideration (Papernot et al., 2017; Athalye et al., 2018). The BPDA attack can be easily adapted to BNNs by learning a SGD-trained NN on a dataset where the label of each image in the training set is replaced by the predictive distribution of the BNN on that image. Once this new NN has been learned, one can run standard gradient-based attacks on this network.

2.1.3. Genetic Algorithm (GA) for Adversarial Attacks

Both ZOO and BPDA involve approximating the gradient of the BNN wrt a given input. This may be limiting for BNNs (see Eqn (2)). In contrast, in this subsection, we consider a genetic algorithm similar to that proposed in (Alzantot et al., 2019) which proceeds in a completely gradient-free manner. Given an input point x^* the genetic algorithm attack (GA) proceeds by first sampling k vectors in an $\epsilon-$ball around x^* from a uniform distribution. These vectors are then each added to the original input to create a set of k candidate adversarial examples. For each candidate, the BNN predictive distribution is queried and each of the k predictions are compared with the original BNN prediction and are scored based on a fitness function. In our case, this fitness function is taken to be the maximizing the loss function (see Eqn 3 in the Appendix). Tournament selection is then used to decide which modifications to keep and which to discard, that is, we randomly draw (with replacement) j pairs of members from the population and select to keep the member with higher

fitness. New members are made by crossing the winners from the tournament selection and some of the resulting set are randomly mutated before repeating these steps. After repeating this process m times (each called a generation), the modification with the highest fitness is returned as the adversarial perturbation. A further description and pseudocode are given in the appendix.

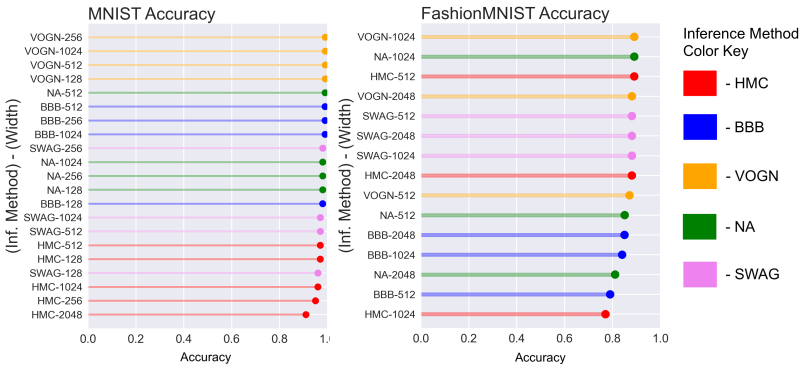

Figure 1: Each point in the graph represents the accuracy of the predictive distribution for approximate posterior. Each value is estimated from 500 test set images. **Left Column:** MNIST test set accuracy is $\geq$95% for each combination of training method and architecture **Centre Column:** FashionMNIST test set accuracy indicates that the approximate BNNs achieve between 80% and 90% accuracy. **Right Column:** Color coordinated key for each inference method that will be used throughout each figure.

3. Empirical Results

We conduct an empirical evaluation of the proposed methods against state-of-the-art gradient-based methods (PGD given here and FGSM in the appendix) for various NN architectures trained with different approximate inference methods, including Hamiltonian Monte Carlo (HMC, Neal et al. (2011)), Bayes by Backprop (BBB, Blundell et al. (2015)), Variational Online Guass Newton (VOGN, Khan et al. (2018)), NoisyAdam (NA, Zhang et al. (2018)), and Stochastic Weight Averaging - Gaussian (SWAG, Maddox et al. (2019)). For each of the BNNs discussed, we select the prior distribution based on the Glorot normal distribution (Sutskever et al., 2013). For MNIST we fix the strength of the adversarial perturbation $\epsilon = 0.1$ and for FashionMNIST we instead consider $\epsilon = 0.05$. For each PGD attack we consider 5 iterations with 1 restart. We note that each iteration of PGD requires estimation of the expectation of the gradient via Monte Carlo integration making this setting as computationally expensive as 1000 iterations of PGD in the deterministic case (assuming 100 steps for the Monte Carlo integration).

We perform our experiments on the MNIST and Fashion MNIST data-sets. We note that in our evaluation, we do not consider bigger and more complex datasets, such as CIFAR-10, because we could not train HMC and BBB on those datasets with good accuracy.

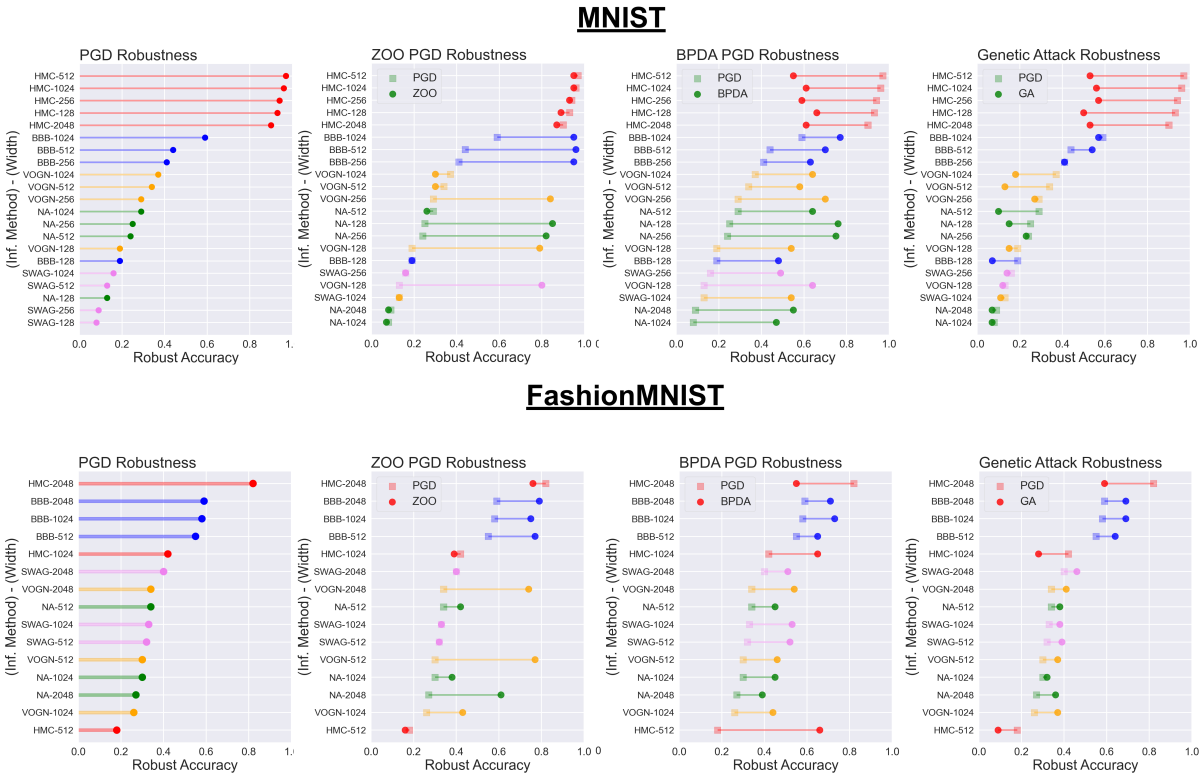

Figure 2: Robust Accuracy of each posterior wrt different attack algorithms. On the far left we plot as dots the robust accuracy of posteriors against PGD. For all other attack algorithms, in order to compare directly with PGD, we plot the PGD performance as a square and draw a line to the robust accuracy of the posterior against that particular attack (given in the title). **Top Row, left to right:** Robust Accuracy on the MNIST dataset for each approximate Bayesian inference method against PGD, PGD with ZOO, PGD with BPDA, and GA. We observe that in all cases GA outperforms other methods. **Top Row, left to right:** Robust Accuracy on the FashionMNIST dataset for each approximate Bayesian inference method against PGD, PGD with ZOO, PGD with BPDA, and GA. We observe that GA outperforms other attacks for HMC, while for other training methods PGD obtains the best performances.

Evaluation of Robust Accuracy on MNIST We start our evaluation with the MNIST dataset. In Figure 1 we report the accuracy of the various NNs on clean data. For this evaluation we consider networks with a single hidden layer of varying width (from 128 to 2048) and we observe that for all the models we obtain an accuracy $\geq 95\%$.

In Figure 2 (top row) we report the computed robustness to adversarial attacks for all the various networks and methods. The robustness is measured with the 'Robust Accuracy', which is a standard measure of robustness (Szegedy et al., 2013) defined as the percentage of test points that remain successfully classified after a given attack (i.e., the smaller the

Robust Accuracy, the less robust the network). From Eqn (2) we expect that, at least for HMC, which is known to produce asymptotically exact samples from the posterior, all the networks are robust to gradient-based attacks. This is indeed the case. However, GA which does not rely on the gradient, is up to 40% more effective in these cases. Note that the obtained values of robustness are still considerably higher than those commonly reported for SGD-trained NNs (Carbone et al., 2020).

For other approximate inference methods, GA is still more effective than PGD. However, we observe that the more approximated is the posterior computation, the more effective are gradient-based attacks. In fact, for all SWAG-trained networks (which in the degenerate case is exactly SGD-training), the robust accuracy to PGD drops from the $\sim 95\%$ of HMC to $\sim 10\%$ (i.e. random guessing). Finally, we observe that, in the case of HMC, ZOO performs comparably with PGD, while BPDA is considerably better and PGD, even if less effective than EA. This is due to the fact that at no point during the optimization does BPDA query the posterior predictive distribution of the BNN under consideration. This again confirms that for BNNs finding adversarial attacks by relying on the loss gradient can be misleading.

Evaluation of Robust Accuracy on Fashion MNIST We continue our evaluation with the Fashion MNIST dataset. In Figure 1 (centre column) we report the accuracy of the various trained NNs. For this evaluation we consider networks with 2 hidden layers with same number of neurons for each layer and of varying width (512, 1024, and 2048).

In Figure 2 (bottom row) we report the robust accuracy for all the various networks and methods. We observe that HMC and BBB trained networks are able to achieve notable success in resisting adversarial examples. Similar to what is observed in MNIST, we see that for HMC trained networks, GA is the most effective algorithm, showing up to 30% improvements over PGD attacks. This again confirms how for BNNs gradient-based attacks can be surprisingly ineffective compared to black-box optimization methods. However, when we move to other approximate inference methods we see that these offer less protection against gradient-based attacks. In fact, in these cases PGD outperforms all the other methods with GA obtaining similar, but slightly worse, results. As already noticed in (Carbone et al., 2020; Gal and Smith, 2018), this may be due to the fact that Variational Inference (VI) methods, for larger and more complex datasets, may tend to poorly approximate the uncertainty and lead to a posterior that is far from the true distribution (Cardelli et al., 2019a). As a consequence, the loss gradient may become meaningful and the resulting networks fragile to gradient-based attacks such as PGD.

4. Conclusion

We investigated gradient-free optimization algorithms for finding adversarial examples on BNNs. In an empirical evaluation on the MNIST and FashionMNIST datasets, we showed that gradient-free algorithms can be more effective than gradient-based methods, especially for NNs trained with approximate inference methods that compute a more accurate approximation of the true posterior.

References

Moustafa Alzantot, Yash Sharma, Supriyo Chakraborty, Huan Zhang, Cho-Jui Hsieh, and Mani B Srivastava. Genattack: Practical black-box attacks with gradient-free optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1111–1119, 2019.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

Artur Bekasov and Iain Murray. Bayesian adversarial spheres: Bayesian inference and adversarial examples in a noiseless setting. *arXiv preprint arXiv:1811.12335*, 2018.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

G. Carbone, M. Wicker, L. Laurenti, A. Patane, L. Bortolussi, and G. Sanguinetti. Robustness of bayesian neural networks to gradient-based attacks. *NeurIPS*, 2020.

Luca Cardelli, Marta Kwiatkowska, Luca Laurenti, Nicola Paoletti, Andrea Patane, and Matthew Wicker. Statistical guarantees for the robustness of bayesian neural networks. *IJCAI*, 2019a.

Luca Cardelli, Marta Kwiatkowska, Luca Laurenti, and Andrea Patane. Robustness guarantees for bayesian inference with gaussian processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7759–7768, 2019b.

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 15–26, 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 J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. *arXiv preprint arXiv:1804.08598*, 2018.

Mohammad Emtiyaz Khan, Didrik Nielsen, Voot Tangkaratt, Wu Lin, Yarin Gal, and Akash Srivastava. Fast and scalable bayesian deep learning by weight-perturbation in adam. *arXiv preprint arXiv:1806.04854*, 2018.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Xuanqing Liu, Yao Li, Chongruo Wu, and Cho-Jui Hsieh. Adv-bnn: Improved adversarial defense through robust bayesian neural network. *arXiv preprint arXiv:1810.01279*, 2018.

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. *Advances in Neural Information Processing Systems*, 32:13153–13164, 2019.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

Rhiannon Michelmore, Matthew Wicker, Luca Laurenti, Luca Cardelli, Yarin Gal, and Marta Kwiatkowska. Uncertainty quantification with statistical guarantees in end-to-end autonomous driving control. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7344–7350, 2020.

Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

Nitasha Soni and Tapas Kumar. Study of various mutation operators in genetic algorithms, 2014.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Dr. Anantkumar Umbarkar and P. Sheth. Crossover operators in genetic algorithms: A review. *ICTACT Journal on Soft Computing (Volume: 6 , Issue: 1)*, 6, 10 2015. doi: 10.21917/ijsc.2015.0150.

Matthew Wicker, Xiaowei Huang, and Marta Kwiatkowska. Feature-guided black-box safety testing of deep neural networks. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 408–426. Springer, 2018.

Matthew Wicker, Luca Laurenti, Andrea Patane, and Marta Kwiatkowska. Probabilistic safety for bayesian neural networks. *UAI*, 2020.

Guodong Zhang, Shengyang Sun, David Duvenaud, and Roger Grosse. Noisy natural gradient as variational inference. In *International Conference on Machine Learning*, pages 5852–5861, 2018.

Jinghui Zhong, Xiaomin Hu, Min Gu, and Jun Zhang. Comparison of performance between different selection strategies on simple genetic algorithms.

Appendix A. FGSM Results

In Figure 3 we report the attack results for the investigated attacks on the studied posteriors with FGSM.

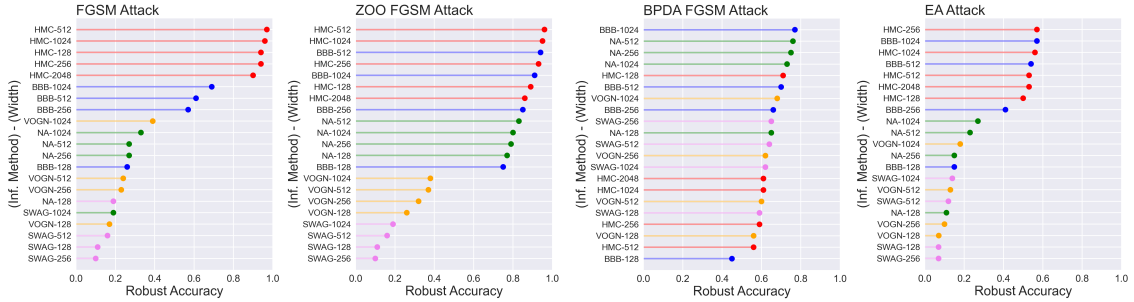

Figure 3: Robust Accuracy of each posterior wrt different attack algorithms. On the far left we plot as dots the robust accuracy of posteriors against FGSM. **Left to Right:** Robust Accuracy on the MNIST dataset for each approximate Bayesian inference method against FGSM, FGSM with ZOO approximate gradients, FGSM with BPDA approximate gradients, GA. We observe that in all cases GA outperforms other methods.

Appendix B. Genetic Attack Details

The Genetic Attack (GA) follows the conventional structure of a genetic algorithm:

1. Initialize a population,
2. compute the fitness of each member of the current population,
3. (selection) select certain members of the current population to be parents for the next population,
4. (crossover) cross the selected parents to form children,
5. (mutation) randomly modify the children and collect them to form the next population, then

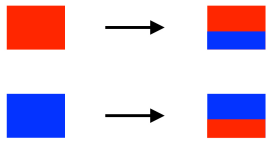

Figure 4: Horizontal crossover.

6. repeat steps 2–4 for multiple generations until a stopping condition is reached.

The goal of a genetic algorithm is to increase the overall fitness of the population. In our case, the population consists of a collection of N perturbation vectors which we may add to an image x to turn it into an adversarial image. We define the fitness at class c of a member m of the population to be the negative logarithm of the cth coordinate of the BNN's prediction for $x + m$ after clipping to make every coordinate in $[0, 1]$.

$$\text{Fitness}(m, c) \coloneqq -\ln\left((\langle f^{\mathbf{w}}(\text{clip}(x+m))\rangle_{p(\mathbf{w}|D)})_c\right), \tag{3}$$

The fitness of m at thew correct class measures how adversarial is. If the $\text{Fitness}(m, c)$ is high, then the BNN predicts that $x + m$ belongs to class c with low probability.

GeneticAttack initializes the population as N random $\{-\varepsilon, \varepsilon\}$-valued vectors with the same dimensions as the input space. Such vectors are the largest perturbations allowed by our constraint, so intuitively they have the best chance at having a high fitness.

Algorithm B is our selection operator. It is a standard operator in genetic algorithms called tournament selection Zhong et al., and we use a tournament size of two. We randomly draw (with replacement) N pairs of members from the population and select from each pair the member with a higher fitness.

Algorithm B is our crossover operation. It is an adaptation of the standard single-point crossover Umbarkar and Sheth (2015) to two-dimensional inputs. Given two parent images, we randomly select a location to cross the parents horizontally or vertically. Figure 4 illustrates horizontal crossover.

Algorithm B is our mutation operator. It decides to mutate each child with probability R, and mutation consists of flipping the sign on precisely one randomly chosen pixel of that child. Our mutation operator might be considered a sparse form of flip mutation Soni and Kumar (2014). In the case of MNIST and Fashion MNIST, we are flipping single coordinates. In the case of CIFAR10 we are flipping a triple corresponding to perturbations in a pixel's RGB values.

GeneticAttack stops after G iterations, and it returns the original image x plus the population member with the highest fitness.

Algorithm 1 Genetic Attack

Data: BNN, original image x, correct class c, number of generations G, population size N, mutation rate R, perturbation size ε.
Result: perturbed image $x*$ (hopefully adversarial)

```
/* Initialize population and fitness.                                  */
 for i ← 1, ..., N do
 |   P_i ← random {−ε, ε}-valued vector
end

for i ← 1, ..., N do
 |  F_i ← Fitness(P_i, c)
end
/* Update population and fitness.                                      */
for g ← 1, ..., G do
 |  P ← Mutate(Cross(Select(P, F)), R) for i ← 1, ..., N do
 |   |  F_i ← Fitness(P_i, c)
 |  end
end
return x + P_{arg max(F)}
```

Algorithm 2 Select

Data: Population P, fitnesses F.
Result: Array of parent perturbations.

```
for i ← 1, ..., size(P) do
 |  select j, k uniformly from {1, ..., size(P)} if F_j > F_k then
 |   |  M_i ← P_j
 |  else
 |   |  M_i ← P_k
 |  end
end
return M
```

Algorithm 3 Cross

Data: Array M of parent perturbations.
Result: Array of child perturbations.

```
for i ← 2, 4, 6, ..., size(M) do
 |  p ← P_{i−1}
 |  q ← P_i
 |  select t uniformly from {0, 1, ..., image side length}
 |  select u uniformly from {1, 2}  if u = 1 then
 |   |  c, d ← cross p, q horizontally at t (Fig. 4)
 |  else
 |   |  c, d ← cross p, q vertically
 |  end
 |  C_{i−1} ← c  C_i ← d
end
return C
```

Algorithm 4 Mutate

Data: Array C of child perturbations, mutation rate R.
Result: Array of mutated

```
for i ← 1, ..., size(C) do
    if r ~ U(0,1) < R then
        c ← C_i
        select pixel j uniformly randomly
        c_j ← −c_j
    end
end
return C
```