Improving Bayesian Neural Networks by Adversarial Sampling: A Replication Study

Farog Bayat, Velin Ekupov, Evgenia Karnavou
f.b.bayat@tilburguniversity.edu, v.s.ekupov@tilburguniversity.edu, e.karnavou@tilburguniversity.edu

Abstract

This is a replications study of a paper by Zhang et al. where the authors propose an improvement on Bayesian Neural Networks by making use of Adversarial Sampling [11]. They suggest that the randomness of sampling in these networks results in errors during training and poor performance of some sampled models during testing. This is why they propose training these networks with Adversarial Distribution. Since calculating Adversarial Distribution is difficult, they propose using Adversarial Sampling as a practical approximation. We replicate their conducted experiments on multiple network structures and datasets, including CIFAR-10 and CIFAR-100. However, our replication study does not confirm the theoretical analysis and the effectiveness of Adversarial Sampling in improving model performance, likely due to experimental constraints i.e. using only 20 epochs instead of 200. Additionally, we intended to test the models on a new dataset, TinyImageNet.

Introduction

The target replication technique for this study is the ”Improving Bayesian Neural Networks by Adversarial Sampling” paper by Zhang et al. [11]. The paper proposes a novel adversarial sampling method for improving the accuracy and uncertainty estimation of Bayesian neural networks (BNNs). The method generates samples in regions where the model’s posterior distribution is highly concentrated, and it is designed to sample adversarial examples that challenge the model’s decision boundaries.

The internal deep perceptron network architecture used in the paper consists of a feedforward neural network with multiple hidden layers. The authors use dropout regularization and batch normalization to prevent overfitting and improve training performance. The architecture also includes a layer for Bayesian inference, which allows for the estimation of the model’s uncertainty.

Our replication study was conducted using the same and similar datasets, methods, and hyperparameters as the original paper. The CIFAR-10 and CIFAR-100 datasets were used for training and testing the model, as well as the Tiny ImageNet dataset. The models were tested using Google Colab in order to use their provided GPU, and the proposed adversarial sampling method was incorporated into the training process. The replication study consisted of several experiments, including evaluating the performance of the model with and without the adversarial sampling method, and using another similar dataset to test the authors’ proposed solution. The performance of the model was evaluated using various metrics, including accuracy, precision, and losses to determine the quality of their proposed models.

We did not find evidence that the proposed adversarial sampling method improved the overall precision of the BNN models. However, it is important to note that our results may not be entirely representative of the situation due to the fact that we trained the models for only 20 epochs instead of 200 epochs, as was done in the original paper. This was due to time and resource constraints. As a result, we cannot rely on our results to draw conclusions about the effectiveness of the adversarial sampling method in improving the performance of Bayesian neural networks.

Further research is needed to validate the results of the original paper, including testing the robustness of the method and assessing the generalizability of the models. Therefore, we suggest that future studies replicate the original experiments using longer training periods and a larger range of hyperparameters to better assess the potential of the adversarial sampling method for improving BNN performance.

The remaining sections of the paper are organized as follows: The next section provides the theoretical background necessary to comprehend the techniques used in the replicated work. This section summarizes the fundamental knowledge and insights related to the paper. The following section, the target study overview, offers a more detailed description of the techniques, study context, methodology, algorithms, and dataset of the targeted study. The replication study definition explains the structure and setup of the replication study that we conducted, and we present some visualizations of our replication results. We also discuss the constraints we encountered in terms of feasibility, which may have impacted the effectiveness and reliability of our paper. Finally, in the last section, we conclude our paper and reflect on the lessons we learned.

Background

Bayesian neural networks (BNNs) differ from deterministic deep neural networks (DNNs) in that they aim to find the posterior distribution of the model parameters instead of a single point estimate [2]. This allows for uncertainty estimation and can lead to better generalization performance, especially in situations where there is limited data or high noise.

There are two main methods used to train BNNs: Variational Inference (VI) [5] and Markov Chain Monte Carlo (MCMC) [3]. VI is computationally more efficient than MCMC and has gained popularity in recent years. However, many existing works fail to achieve practicality, and there are few real-world applications of BNNs in industry.

To address this issue, researchers have proposed various methods to improve the performance and practicality of BNNs. Wenzel et al. showed that cooling the posterior with a temperature parameter improves generalization performance, but it can also lead to the posterior distribution becoming more concentrated on the set of point estimates, effectively turning the BNN into a DNN [10]. There are also studies that proposed using pretrained DNNs to establish priors and initialize model parameters, leading to improved performance [6]. However, the underlying reasons for the effectiveness of these methods are not well-understood, and there is still room for further improvement in the field of BNNs.

One of the major challenges that need to be addressed before safety-critical applications can be widely deployed is the vulnerability of deep learning models to adversarial attacks. A recent study by Bortolussi et al. offers a new perspective on the robustness of Bayesian Neural Networks to adversarial attacks [1]. They analyze the geometry of adversarial attacks in the large-data, overparameterized limit for BNNs. The study shows that vulnerability to gradient-based attacks in this limit is due to degeneracy in the data distribution, which occurs when the data lies on a lower-dimensional submanifold of the ambient space.

Furthermore, the study provides evidence that BNNs can be robust to gradient-based adversarial attacks in this limit [1]. The authors prove that the expected gradient of the loss with respect to the BNN posterior distribution is vanishing, even when each neural network sampled from the posterior is vulnerable to gradient-based attacks. Experimental results with BNNs trained with Hamiltonian Monte Carlo and Variational Inference support this line of argument, demonstrating that BNNs can exhibit high accuracy on clean data and robustness to both gradient-based and gradient-free-based adversarial attacks. Overall, this study suggests that BNNs can be a promising approach for developing deep learning models that are robust to adversarial attacks.

Target Study Overview

The target study of this replication report is ”Improving Bayesian Neural Networks by Adversarial Sampling” by Zhang et al. [11]. In this section, we will give an overview of the important aspects of this study in more detail.

Technique foundations The study builds upon the Bayesian neural network (BNN) framework, which is a type of neural network that incorporates Bayesian inference to estimate uncertainty in model predictions. The study proposes a novel adversarial sampling technique for improving the accuracy and uncertainty estimation of BNNs.

Technique intervals The proposed adversarial sampling method generates samples in regions where the model's posterior distribution is highly concentrated. This is achieved by minimizing a loss function that maximizes the model's predictive entropy and simultaneously minimizes the Wasserstein distance between the true and generated samples. The method is designed to sample adversarial examples that challenge the model's decision boundaries and improve its generalization performance.

Study context and targeted domain The study targets the domain of machine learning, specifically Bayesian neural networks, which are widely used in applications such as image classification, natural language processing, and speech recognition. The proposed adversarial sampling method is designed to improve the accuracy and uncertainty estimation of BNNs in image classification tasks.

Basic techniques inherited by the study The study inherits its basic techniques such as Bayesian inference, neural networks, and optimization methods. The authors use PyTorch as the framework for implementing the proposed method.

Validation approach and methodology The proposed adversarial sampling method is evaluated on two benchmark datasets, CIFAR-10, and CIFAR-100, which are widely used in image classification tasks [7]. The evaluation is conducted using various metrics, including accuracy, calibration error, and negative log-likelihood. The authors compare the performance of the proposed method with several state-of-the-art methods, including Monte Carlo Dropout and Deep Ensembles.

Algorithms involved in training, testing, and validation The proposed adversarial sampling method is implemented using PyTorch. The authors use stochastic gradient descent with a learning rate schedule to train the model. The model is trained for a fixed number of epochs, and the best hyperparameters are selected based on the validation set performance. The testing and validation are performed on separate datasets, and the model's performance is evaluated using various metrics.

Evaluation dataset, overview and explanation The targeted study uses two benchmark datasets, CIFAR-10, and CIFAR-100, for evaluating the proposed method. CIFAR-10 is a dataset of 60,000 32x32 color images in 10 classes, while CIFAR-100 is a dataset of 60,000 32x32 color images in 100 classes [7]. The datasets are split into training, validation, and testing sets, and the model's performance is evaluated on the testing set. The authors provide a detailed description of the datasets and their preprocessing steps in the paper.

Study limitations and replication problem statement The study has some limitations, including the use of a fixed number of epochs for training and the limited exploration of hyperparameter settings. The replication problem statement that we are addressing in this report is to reproduce the results reported in the paper using both the same and similar datasets, methods, and hyperparameters, and to explore the robustness of the proposed method to variations in these factors. We want to evaluate the results by training the proposed models on another similar dataset. The goal is to reproduce the results reported in this study by using an operational replication approach.

Replication Study Definition

The goal of the replication study performed is to evaluate the work of [11], and test whether their methods and models yield similar performance on similar datasets. Specifically, while the authors of the replicated paper tested their proposed solution on CIFAR-10 and CIFAR-100, we trained their models on the Tiny ImageNet, which consists of a down sample of the original ImageNet dataset. We determine the generalizability of the authors' results with respect to the same metrics used in their paper, namely the accuracy. Additionally, we present the losses as well as the precision of the models as additional metrics to determine the quality of performance of the proposed models.

Importance of BNNs and Replication Study

To accurately justify the need of BNNs, one has to consider the weaknesses of deterministic NNs. Specifically, NNs have been characterized as 'statistical black boxes', meaning that the interpretation of the decision-making process of NNs and the internal workings of such networks are difficult to explain using traditional methods [4].

This lack of interpretation, along with the sometimes over-confident estimates, renders NNs somewhat unsuitable for high risk domains such as medical diagnostics and autonomous vehicles [4]. The need to mitigate the threat of uncertainty noted in Neural Networks led to the development of BNNs. Utilizing Bayesian Statistics, BNNs provide a natural way to reason about uncertainty in predictions, and they can also provide information about the decision-making process. Therefore, their main advantage is that they offer uncertainty estimations about predictions, an ability that is particularly useful in fields where determining uncertainty is important, such as medical diagnosis or financial forecasting. However, like all NNs, BNNs suffer the consequences of the inherent randomness that defines such machine learning approaches.

In general, Neural Networks have a characteristic aspect of randomness. This stochasticity concerns various attributes of the network's structure, one of those being the initialisation of the weights and biases. Since the values for these parameters are drawn randomly from a distribution, the resulting performance of a network may vary in relation to these different initialisations. Indeed, [9] has already demonstrated how this type of randomness might effect a network's performance during the training process. Now, in the case of BNNs, the authors of the replicated study clearly demonstrated in their paper how randomness in the initialization process of the Bayesian Neural Network can cause some parameters to be updated wrongly. To mitigate the thread of randomness, which leads to poor performance, the authors utilized Adversarial Training to attempt to improve the performance of BNNs. Indeed, the authors reported better performance when Adversarial Sampling was used, for all the models, and both datasets.

It is obvious how optimizing the performance of such models help practitioners from all sorts of fields to make more trustworthy and accurate predictions. Therefore, the implications of our replication report could have an impact on practitioners who seek to make informed and certain predictions, as well as researchers. For the latter, BNNs provide the opportunity to analyse the uncertainty estimates of such models, and that would help in identifying data points that cause fluctuations in the model performance, and therefore either improve the data, or optimize the model architecture so that it is more robust.

Replication Approach and Design Decisions

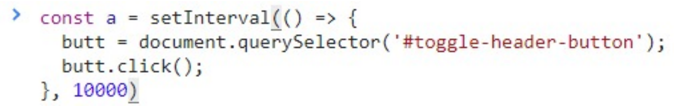
In this paper, we have adopted an operation replications approach, for the following reasons:

1. The authors hypothesized in their paper that Bayesian Neural Networks trained with Adversarial Sampling would enjoy a higher performance to those trained without it. Indeed, their results clearly demonstrate that that's the case, but specifically for the datasets the authors utilized for training and testing the neural network. Specifically, when Adversarial Training was used for their Bayesian Neural Network, the models had higher accuracy for both the CIFAR-10 and CIFAR-100 datasets. However, both these datasets are considered simple, and we wanted to test the performance of the proposed solution on a dataset that requires more complex models for image recognition tasks. This is why we decided to replicate the methodologies and models used in the replicated paper, but on the Tiny ImageNet dataset, which is a subset of the original ImageNet.

Design Decisions & Datasets In order to ensure that our replication approach was accurate, we performed the exact same experiment, adopting the same experimental settings as those in our referenced work. More specifically, we used the same sampling techniques, but a different dataset.

The Tiny ImageNet contains 100,000 images of 200 classes (500 for each class), downsized to 64x64 coloured images. Each class had 500 training images, 50 validation images and 50 test images. The CIFAR-10 dataset consists of 60,000 32x32 colour images in 10 classes, with 6,000 images per class. The 10 classes are: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The CIFAR-100 dataset is similar to CIFAR-10, but has 100 classes, with each class containing 600 images. The 100 classes are grouped into 20 superclasses, with each superclass containing 5 subclasses. The images in Tiny ImageNet are of varying sizes and resolutions, with some images containing multiple objects or scenes. Tiny ImageNet is more complex than CIFAR-10 and CIFAR-100 because of the larger number of object classes, greater variability in object appearance, and the presence of multiple objects and scenes in many images.

Besides the difference in datasets, the exact same models and configurations were used for training, testing and evaluating. The reason for this is to determine whether the results of the original study are replicable under a different, more complex dataset. Additionally, the measures used to indicate performance of the algorithm is mainly the evaluation and training precision, as well as the loss. Finally, the mean accuracy over the evaluation and testing dataset was computed over 20 epochs, and not 200 as done in the referenced work. The reason for that is that the entire algorithms takes (literally) ages to train. Specifically, 100 epochs take more than 15 hours to run. Another difficulty we came across was the fact that in order to run the experiment, it was required for the computer to have GPU. Since for two of our members running the code locally was not an option, we utilized Google Colab, and used their provided GPU resources that were otherwise unavailable. However, due to the long runtime of the algorithm, and the fact that Google Colab is interactive and needs some sort of mouse interaction from the user every twenty minutes, we implemented a Java script (Figure 1) that was used to automate a 'fake' interaction with the webpage.



```
> const a = setInterval(() => {
    butt = document.querySelector('#toggle-header-button');
    butt.click();
  }, 10000)
```

Figure 1: **Java Script for automatic interaction with the Google Colab Console.**

However, even with the interaction restriction handled, Google Colab offers limited GPU credits for users that are not paying a subscribed fee in order to access more resources. Therefore, another issue we encountered had to do with the 'Credits out' error that Colab would give as the execution continued. This effectively halted execution, and we had to wait another day for GPU credits to be available again, and therefore to continue running model configurations. That certainly led to a lot of lost time and frustration, since we would sometimes lose data of the runs, and were not able to continue running anything for 24 hours. Additionally, we initially wanted to train the models for 100 epochs, but after the first 47 the GPU limit ran out, so training for more than 47 epochs was not feasible in our case unfortunately. We would also like to mention that 47 epochs has a runtime of 8 hours, so training our models with even 40 epochs was not feasible considering the deadlines timeline.

Results

Before discussing the results, it is important to mention that we set the epochs argument from 200 to 20, for time efficiency reasons. Specifically, in our computers, the overall experiment for a **single configuration** takes about 2.5 hours. Since we wanted to test the performance for all different configurations, we decided it would be a good trade-off to explore different models, but have biased results due to little iterations over the datasets. Considering that, in total we had 3 modes, with two configurations each, and three datasets, we thought that 20 epochs would give an idea of the overall performance. We understand how this decision may be suboptimal since 20 epochs are not enough to get an accurate performance overview and metrics, however we did the best with the time and computational resources we had.

CIFAR-10

In Table 1, we see the best resulting precision scores of the different configurations for the *testing* dataset, for CIFAR-10. In the Table, AS is shorthand for Adversarial Sampling.

Model	Precision with AS	Precision without AS
Resnet20	64.64	84.59
Resnet56	65.16	86.95
Vgg	77.7	85.82

Table 1: **Best resulting precision scores on the test set for the CIFAR-10 dataset.**

Here it is noteworthy that, while in the replicated work all models achieve higher performance with Adversarial Sampling, our results show the opposite. Specifically, the runtimes for training models with Adversarial Sampling were significantly higher than when training without AS. As we can see from the precision scores illustrated above, all models enjoy higher precision when trained without adversarial sampling. We are aware that a comparison is inaccurate, since 20 epochs are considered too little to get accurate results, and additionally, the authors of the replicated paper trained their models for 200 epochs.

In Figure 2, we can see the precision scores for each epoch, for each of the three models: resnet20, resnet56, and vgg; trained with and without Adversarial Sampling for the CIFAR10 dataset. Again, it is obvious that all models perform better (at least initially) without adversarial sampling.

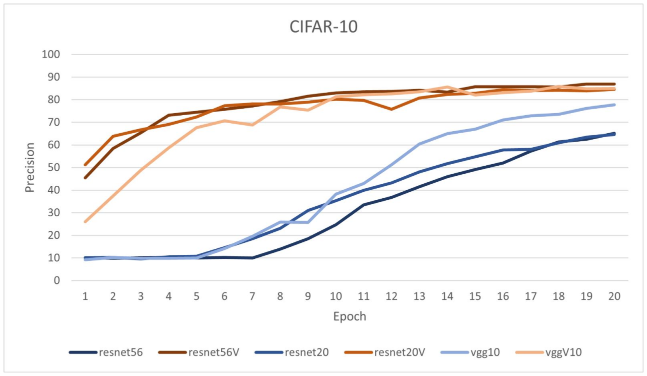


Figure 2: **Plots of the epoch (testing) precisions for each of our three models (resnet20, resnet56 and vgg), with and without adversarial sampling. Models that end with V denote the lack of Adversarial Sampling.**

Why models with Adversarial Sampling perform worse Looking at our results initially, we were perplexed, since in the replicated paper models with Adversarial Sampling always achieved better results. After some research, we reached the conclusion that the reason our results show the opposite is due to the very small amount of epochs.

Generally, models trained with adversarial sampling can perform worse than models trained without it in the first few epochs because adversarial samples are designed to be difficult for the model to classify correctly. When such difficult samples are included in the training set, the model is forced to learn to classify them correctly from the beginning, which can initially slow down the learning process and result in lower accuracy during the early epochs.

In addition, the process of generating adversarial samples can be computationally expensive, and the added complexity can make training with adversarial sampling take longer to converge. This justifies the significantly longer training time we had when training our models with Adversarial Sampling. As a result, models may need more time to learn from the adversarial samples and improve performance.

However, research has demonstrated that the long-term benefits of training with adversarial sampling typically outweigh the initial costs. By training with adversarial samples, the model learns to be more robust to small perturbations in the input data, which can help it perform better on unseen data. For example, Xiaoan Ding, et al. [8] investigated the effects of adversarial training on semi-supervised text classification and reported that while the initial performance of models trained with adversarial examples was lower than those trained without them, the final performance was significantly better. Therefore, we conclude that the difference in performance with Adversarial Sampling and without it can be appointed to the small number of epochs, which in our case were chosen for time and resources efficiency.

CIFAR-100

Our results for the CIFAR-100 dataset were strikingly similar. Again, the models trained with adversarial sampling performed worse than those that were trained without it, for our 20 epochs. We can safely assume that the reason for that is the same as with the CIFAR-10 dataset. Our only noteworthy difference is that for CIFAR-100 all models, both with AS and without it, performed notably worse for this dataset.

As is evident from Table 2, the resulting precisions, both for models with AS and without it, are much lower that the ones produced for the CIFAR-10 dataset.

Model	Precision with AS	Precision without AS
Resnet20	30.25	55.73
Resnet56	27.91	54.97
Vgg	15.5	40.89

Table 2: **Best resulting precision scores on the test set for the CIFAR-100 dataset.**

Figure 3 presents the precision scores for the CIFAR-100 dataset.

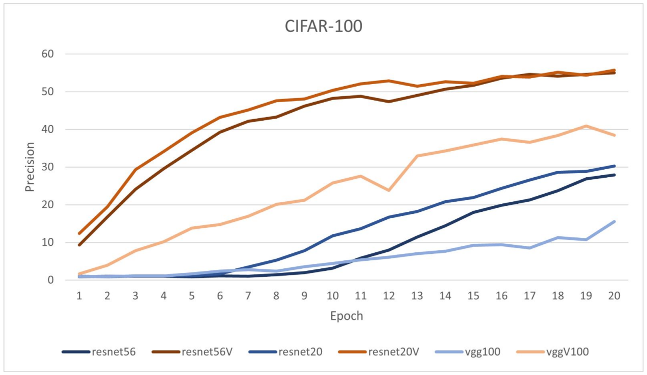


Figure 3: **Plots of the epoch (testing) precisions for each of our three models (resnet20, resnet56 and vgg), with and without adversarial sampling. Models that end with V denote the lack of Adversarial Sampling.**

We believe that that is a consequence of the increased complexity of the CIFAR-100 dataset, that contained more classes and more images.

Tiny ImageNet

Unfortunately, we do not have any metrics and results to show for the Tiny ImageNet dataset. While we did train the models on the dataset, again for 20 epochs each, we unfortunately were not able to retrieve the precisions per epoch, like we did for the previous models. The reasons for that is that the team member running those configurations did not save the output logs in a .txt file.

Admittedly, none of us did in the beginning. We were unfortunately too late to realise that the replicated paper's result was just a .pt pyTorch file, that contained the weights of each trained model. Therefore the execution log (that contained the precision scores of each epoch) was nowhere to be found. While two team member were able to rerun the code to get the log files and insert them into .txt files, training with the Tiny ImageNet again was not feasible in the time left, since its runtime is considerably longer.

Threats to Validity

In this section, we discuss the threats that might have affected the effectiveness and reliability of our replication study.

First of all, this replications study is incomplete, since we were reaching for an operational type of approach, but ended up with a literal replication, which in itself was limited. By that, we mean that it wasn't an actual literal replication, since the epochs we trained our model for were 20, and not 200 as the original paper.

We understand how that might have affected the outcomes, and we believe that the results we got were not representative of the situation. Specifically, we believe that, had we had the time to train the models with the full 200 epochs, the models with adversarial sampling would have performed better than those without it. Therefore, we believe our replication study to be inconclusive.

Conclusions

This report presents our attempt to replicate the study conducted by Zhang et al. [11] by testing various Bayesian Neural Networks with and without Adversarial Sampling to optimize their performance. To achieve this, we provided an overview of the theoretical foundation required to comprehend and replicate the primary techniques utilized in the original study, followed by conducting the same experiments as the original study. We identified several concerns and limitations of our study, such as the extensive computational resources required to run different model configurations and our transition from an operational replication approach to a literal replication due to time constraints and lack of insight.

Despite the challenges we faced, we made every effort to allocate our resources optimally and complete the study within the given timeframe. Overall, our literal replication provides insights into the benefits and challenges of Adversarial Sampling and serves as a potential precursor to improving Bayesian Neural Networks. We expect that our findings will contribute to the growing body of literature on BNNs and emphasize their future significance in classification tasks

References

[1] Luca Bortolussi et al. *On the Robustness of Bayesian Neural Networks to Adversarial Attacks*. 2022. arXiv: 2207.06154 [cs.LG].

[2] Wray L. Buntine and Andreas S. Weigend. "Bayesian Back-Propagation". In: *Complex Syst.* 5 (1991).

[3] Tianqi Chen, Emily B. Fox, and Carlos Guestrin. *Stochastic Gradient Hamiltonian Monte Carlo*. 2014. arXiv: 1402.4102 [stat.ME].

[4] Ethan Goan and Clinton Fookes. "Bayesian neural networks: An introduction and survey". In: *Case Studies in Applied Bayesian Data Science: CIRM Jean-Morlet Chair, Fall 2018* (2020), pp. 45–87.

[5] Alex Graves. "Practical Variational Inference for Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor et al. Vol. 24. Curran Associates, Inc., 2011. URL: https://proceedings.neurips.cc/paper_files/paper/2011/file/7eb3c8be3d411e8ebfab08eba5f49632-Paper.pdf.

[6] Ranganath Krishnan, Mahesh Subedar, and Omesh Tickoo. *Specifying Weight Priors in Bayesian Deep Neural Networks with Empirical Bayes*. 2019. arXiv: 1906.05323 [cs.NE].

[7] Alex Krizhevsky and Geoffrey Hinton. *Learning multiple layers of features from tiny images*. Tech. rep. 0. Toronto, Ontario: University of Toronto, 2009.

[8] Takeru Miyato, Andrew M Dai, and Ian Goodfellow. "Adversarial training methods for semi-supervised text classification". In: *arXiv preprint arXiv:1605.07725* (2016).

[9] Wouter F Schmidt et al. "Initializations, back-propagation and generalization of feed-forward classifiers". In: *IEEE International Conference on neural networks*. IEEE. 1993, pp. 598–604.

[10] Florian Wenzel et al. *How Good is the Bayes Posterior in Deep Neural Networks Really?* 2020. arXiv: 2002.02405 [stat.ML].

[11] Jiaru Zhang et al. "Improving Bayesian Neural Networks by Adversarial Sampling". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.9 (June 2022), pp. 10110–10117. DOI: 10.1609/aaai.v36i9.21250. URL: https://ojs.aaai.org/index.php/AAAI/article/view/21250.