Revista Colombiana de Estadística - Applied Statistics
July 2025, volume 48, issue 2, pp. 183–200
http://doi.org/10.15446/rce.v48n2.116416

Adversarial Image Detection Based on Bayesian Neural Layers

Detección de imágenes adversarias basada en capas neuronales Bayesianas

Diego Nicolás Ávila Moreno[1,a], Héctor J. Hortúa[1,b], Andrés Mora-Valencia[2,c]

[1]Laboratorio de Inteligencia Artificial (SavIA Lab), Grupo Signos, Departamento de Matemáticas, Universidad El Bosque, Bogotá, Colombia

[2]School of Management, Universidad de los Andes, Bogotá, Colombia

Abstract

Although Deep Neural Networks (DNNs) have repeatedly shown excellent performance, they are known to be vulnerable to adversarial attacks that contain human-imperceptible perturbations. Multiple adversarial defense strategies have been proposed to alleviate this issue, but they often demonstrate restricted practicability regarding efficiency and handling solely specific attacks. In this paper, we analyze the performance of Bayesian Neural Networks (BNNs) endowed with flexible approximate posterior distribution for detecting adversarial examples. Furthermore, we study how robust the detection method is when Bayesian layers are located at the top or throughout the DNNs to determine the role of the network's hidden layers, and we compare the results with the deterministic ones. We show how BNNs offer a powerful, and practical method of detecting adversarial examples in comparison with deterministic approaches. Finally, we discuss the impact of having well-calibrated models as detectors and how non-gaussian priors enhance the performance of the detection.

***Key words*:** Adversarial attacks; Bayesian neural networks.

Resumen

Aunque las redes neuronales profundas (DNN, por sus siglas en inglés) han demostrado un excelente desempeño, se sabe que son vulnerables a ataques adversarios que contienen perturbaciones imperceptibles para el ser humano. Se han propuesto múltiples estrategias de defensa adversaria para

[a]Graduate student. E-mail: davilam@unbosque.edu.co
[b]Graduate student. E-mail: hhortuao@unbosque.edu.co
[c]Ph.D. E-mail: a.mora262@uniandes.edu.co

mitigar este problema, pero a menudo estas muestran una viabilidad limitada en cuanto a eficiencia y manejo de ataques específicos. En este artículo, analizamos el desempeño de las redes neuronales bayesianas (BNN, por sus siglas en inglés) dotadas de una distribución a posteriori aproximada flexible para detectar ejemplos adversarios. Además, estudiamos la robustez del método de detección cuando las capas bayesianas se ubican en la parte superior o a lo largo de las DNN para determinar el papel de las capas ocultas de la red, y comparamos los resultados con los deterministas. Mostramos cómo las BNN ofrecen un método potente y práctico para detectar ejemplos adversarios en comparación con los enfoques deterministas. Finalmente, analizamos el impacto de contar con modelos bien calibrados como detectores y cómo las distribuciones a priori no gaussianas mejoran el desempeño de la detección.

Palabras clave: Ataques adversarios; Redes neuronales bayesianas.

1. Introduction

Although deep neural networks (DNNs) have demonstrated remarkable achievements, they have also been found to be vulnerable to adversarial examples (Xie et al., 2022; Guesmi et al., 2022). These attacks are generated from imperceptible perturbations applied to the data. When the perturbed image is observed, sometimes is not possible to identify a change in the image context, but still, the networks make undesirable decisions. This problem is even presented in machine learning applications that prioritize safety, such as self-driving cars, aviation control systems, and healthcare systems (Bortsova et al., 2020; Wu et al., 2021; Ben Braiek et al., 2022). Though several methods have been proposed to detect adversarial examples, they still perform poorly on unseen attacks (Wang et al., 2023). Popular explanations for these findings include the fact that adversarial examples lie off the manifold of unperturbed data, residing in regions where the network cannot extrapolate. However, Bayesian neural networks (BNNs) have shown the ability to expose these regions by associating them with higher uncertainty values (Smith & Gal, 2018). Recently, several works have been proposed including BNNs as detectors or defense approaches against adversarial attacks (Li, Tang, Hsieh & Lee, 2021; Deng et al., 2021*a*). However, BNNs require more expertise for practical implementations and more effort for training due to the complexity during their optimization and duplication of the weights (Fortuin et al., 2021). Besides, simple posterior assumptions often give lower performance than their deterministic counterparts and unreliable uncertainty estimates. In terms of terminology, it is still questionable what methods should be employed for transforming deterministic networks into those BNNs, i.e., insert a set of Bayesian layers at either the top of the network (Bayesian Last Layer) or in the entire architecture (García-Farieta et al., 2023). The motivation for exploring both possibilities comes from the fact that intuitively, adding a Bayesian layer at the end of the network, can be viewed as Bayesian linear regression (Mukherjee et al., 2023) with a learnable projected feature space, allowing for a successful balance between scalability and the degree of model-agnosticism (Fiedler & Lucia, 2023; Watson et al., 2021).

Although fully Bayesian networks would demand high computational resources, it has been reported that their Bayesian hidden layers are susceptible to out-of-distribution (OOD) examples that might improve predictive uncertainty estimates (Henning et al., 2021). In this paper, we analyze the behavior and performance of algorithms to detect adversarial examples based on Bayesian Neural Networks inspired in the results reported in Li, Tang, Hsieh & Lee (2021). Due to their stochasticity, BNNs yield stronger robustness guarantees compared to deterministic networks where their design and optimization seem not to be sensitive to data with any style of perturbation. Through several experiments, we found that BNNs achieve better performance in detecting adversarial examples than their deterministic counterpart. We further discuss the impact of the priors over the weights on the performance of adversarial detection. The novelty of this work lies in leveraging BNNs for adversarial detection by exploiting their inherent uncertainty quantification, which helps identify adversarial examples through deviations in predictive distributions. A key advancement is the incorporation of Multiplicative Normalizing Flows (MNF), enabling more flexible posterior approximations that capture complex, non-Gaussian weight distributions, enhancing robustness against adversarial perturbations compared to standard variational inference with restrictive Gaussian assumptions. Furthermore, a comparative analysis of different priors (e.g., Gaussian vs. heavy-tailed) provides insights into how prior choice affects uncertainty estimation, interpretability, and model resilience in adversarial settings.

2. Related Work

There exist multiple works regarding the design of adversarial examples detectors such as RBF-SVM classification proposed by Lu et al. (2017), KNNs approaches implemented by Ma et al. (2018), and distance approaches as measurements for the degree of perturbation aggregated at images as shown in Lee et al. (2018); Cohen et al. (2019); Craighero et al. (2021). Recently, Li, Tang, Hsieh & Lee (2021); Deng et al. (2021*b*); Huck Yang et al. (2022) have proposed the use of Bayesian Neural Network as a novel alternative to detect adversarial attacks. The explanation for which the latter approach seems to work efficiently can be found in Smith & Gal (2018) where the epistemic uncertainty plays a notable role. Regarding epistemic uncertainty (Hüllermeier & Waegeman, 2019) this can be defined as the presence of uncertainty caused by the lack of information or data in this case on a neural network model. On the other hand, adversarial attacks have been categorized depending on their perturbation constraint. Attacks such as the Projected Gradient Method (PGD) (Madry et al., 2017), and the Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014) are l_∞ that fall into the white-box scenario which allows to add limited perturbation to all pixels, while l_2 attacks, such as Carlini and Wagner (CW) (Carlini & Wagner, 2016), and DeepFool (Moosavi-Dezfooli et al., 2015) attack under general decision boundaries. The white-box scenario (Li, Cheng, Hsieh & Lee, 2021) is the situation in which at the time of generating an attack there is complete knowledge of the structure and properties

of the victim neural network. Recently, AutoAttack (Croce & Hein, 2020) has proven to be a more effective attack by combining the Fast Adaptive Boundary Attack (FAB) (Croce & Hein, 2019), AutoPGD (Croce & Hein, 2020) and the Square attack (Andriushchenko et al., 2019).

3. Background

3.1. Adversarial Attacks

Standard DNNs are trained based on the maximum posterior estimation given by Deng et al. (2021*a*),

$$\max_{w} \frac{1}{n} \sum_{i=1}^{n} \log p(y_i|x_i;\omega) + \frac{1}{n} \log p(\omega), \tag{1}$$

where $(x_i, y_i)_{i=1}^{n}$ is the training dataset, $p(y_i|x_i;\omega)$ stands for the predictive distribution. Adversarial attacks can be defined as

$$x_i^{\text{adv}} = x_i + \arg\min_{\delta_i \in \mathcal{S}} \log p(y_i|x_i + \delta_i;\omega) \tag{2}$$

being $\mathcal{S} = \{\delta : ||\delta||_\infty \leq \epsilon\}$ a slightly perturbation balanced via $\epsilon > 0$. The primary objective of adversarial defense is to prevent models from making erroneous predictions on adversarial examples. Adversarial detection techniques aim to identify and filter out adversarial inputs before they affect model decisions, thereby mitigating harmful outcomes (Deng et al., 2021*a*). Yet, achieving robust generalization to unseen attack types and extending these methods beyond image classification remains an unresolved challenge.

3.2. Bayesian Neural Networks

Let $\mathcal{D}$ be a dataset consisting of pairs $\{(x_1, y_1), \ldots (x_n, y_n)\}$, and a neural network with weights w modeled by the conditional probability $p(y|x, w)$ (Abdar et al., 2020; Gal, 2016; Graves, 2011). The posterior distribution, denoted as $p(w|\mathcal{D})$, can be derived using Bayes' law: $p(w|\mathcal{D}) \sim p(\mathcal{D}|w)p(w)$. This expression involves a likelihood function, $p(\mathcal{D}|w)$, which represents the probability of the observed data $\mathcal{D}$ given the weights w, as well as a prior distribution on the weights, denoted as $p(w)$. In the context of neural networks, it is important to note that the direct computation of the posterior is not feasible. To handle this limitation, variational inference (VI) is used to approximate the posterior distribution, via a single distribution $q_\phi(w)$ (Graves, 2011). VI assumes that the true posterior can be approximated by a tractable distribution $q_\phi(w)$. This restricts the flexibility if the true posterior is multimodal or non-Gaussian. VI obtains the best approximation by maximizing the following Evidence Lower Bound (ELBO) to the variational posterior parameters (Zhang et al., 2017)

$$\mathcal{L}(\phi) = \mathcal{E}_{q_\phi(w)} \left[\log p(y|x, w)\right] - \text{KL}(q_\phi(w)||p(w)), \tag{3}$$

where $\mathcal{E}_{q_\phi(w)}$ denotes expectation over parameters w sampled from $q_\phi(w)$. ELBO is a lower bound on the marginal log-likelihood of the dataset $\mathcal{D}$, and assuming that the approximate posterior distribution can be described as a Gaussian mean-field distribution, $q(w|\theta) = \prod_{ij} \mathcal{N}(w; \mu_{ij}, \sigma_{ij}^2)$ where i and j are the indices of the neurons from the previous- and the current layers, respectively, limits the assumptions for high complexity in the weights. This distribution is not flexible enough to capture the real posterior distribution which is much more complex. Then, the accuracy of the model's uncertainty estimates is potentially compromised.

3.2.1. Multiplicative Normalizing Flows (MNF)

Gaussian mean-field distributions are the most commonly utilized family for VI applied to BNNs. Unfortunately, this distribution cannot adequately represent the features of the true posterior. Hence, it is clear that enhancing the complexity of the approximate posterior would yield significant performance improvements. Multiplicative Normalizing Flows (MNFs) have been proposed to efficiently adapt the posterior distributions through the utilization of auxiliary random variables and the normalizing flows (Louizos & Welling, 2017). Mixture normalizing flows (MNFs) suggest that the variational posterior can be mathematically represented as an infinite mixture of distributions

$$q(w|\theta) = \int q(w|z,\theta) q(z|\theta) dz\,, \tag{4}$$

with θ the learnable posterior parameter, and $z \sim q(z|\theta) \equiv q(z)$ the vector with the same dimension of the input layer, which plays the role of an auxiliary latent variable. The latter can increase the flexibility of the variational posterior via normalizing flows. Starting from samples $z_0 \sim q(z_0)$ from fully factorized Gaussians, a rich distribution $z_K \sim q(z_K)$ can be obtained by applying successively invertible f_k-transformations (Jimenez Rezende & Mohamed, 2015)

$$z_K = \mathrm{NF}(z_0) = f_K \circ \cdots \circ f_1(z_0)\,, \tag{5}$$

$$\log q(z_K) = \log q(z_0) - \sum_{k=1}^{K} \log \left| \det \frac{\partial f_k}{\partial z_{k-1}} \right|\,. \tag{6}$$

To handle the intractability of the posterior, Louizos & Welling (2017) suggest to use again Bayes law $q(z_K)q(w|z_K) = q(w)q(z_K|w)$ and introduce a new auxiliary distribution $r(z_K|w,\phi)$ parameterized by ϕ, to approximate the posterior distribution of the original variational parameters $q(z_K|w)$ to further lower the bound of the KL divergence term. Accordingly, the KL-divergence term can be rewritten as follows Louizos & Welling (2017)

$$\begin{aligned} -\mathrm{KL}\big[q(w)\big\|p(w)\big] \geq \mathcal{E}_{q(w,z_K)}\Big[&-\mathrm{KL}\big[q(w|z_K)\big\|p(w)\big] \\ &+ \log q(z_K) + \log r(z_K|w,\phi)\Big]\,. \end{aligned} \tag{7}$$

The first term can be analytically computed since it will be the KL-divergence between two Gaussian distributions, while the second term is computed via the normalizing flow generated by f_K (see Equation 6). Furthermore, the auxiliary posterior term is parameterized by inverse normalizing flows

$$z_0 = \mathrm{NF}^{-1}(z_K) = g_1^{-1} \circ \cdots \circ g_K^{-1}(z_K)\,, \tag{8}$$

and

$$\log r(z_K|w,\phi) = \log r(z_0|w,\phi) + \sum_{k=1}^{K} \log \left| \det \frac{\partial g_k^{-1}}{\partial z_k} \right|\,, \tag{9}$$

where one can parameterize g_K^{-1} as another normalizing flow. A flexible parametrization of the auxiliary posterior can be written as

$$z_0 \sim r(z_K|w,\phi) = \prod_i \mathcal{N}(z_0; \tilde{\mu}_i(w,\phi), \tilde{\sigma}_i^2(w,\phi))\,, \tag{10}$$

where the parameterization of the mean $\tilde{\mu}$, and the variance $\tilde{\sigma}^2$ is carried out by the masked RealNVP (Dinh et al., 2016) as the choice of normalizing flows.

3.3. Model Calibration

In real-world decision-making systems, it is imperative to have the prediction uncertainty of deep learning models. As we discussed earlier BNNs provide the mathematical tools for the quantification of uncertainty. Nevertheless, it tends to be miscalibrated, i.e., the uncertainty does not correspond well to the model error. One way to quantify the miscalibration of neural networks is via the Expected Uncertainty Calibration Error (UCE) (Laves et al., 2019), where the neural network output is partitioned into M bins with equal width and a weighted average of the difference between accuracy and confidence (softmax likelihood) is used

$$UCE = \sum_{m=1}^{M} \frac{|B_m|}{m} |err(B_m) - uncert(B_m)| \tag{11}$$

being

$$err(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbf{1}(\hat{y}_i \neq y), \tag{12}$$

with $\mathbf{1}(\hat{y}_i \neq y) = 1$ and $\mathbf{1}(\hat{y}_i = y) = 0$, and

$$uncert(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} H(p_i), \tag{13}$$

where H stands for the normalized entropy associated with the final probability vector obtained by Montecarlo integration p_i (Laves et al., 2019). A model is perfectly calibrated when its UCE is zero.

4. Methodology

Following the methodology proposed in Li, Tang, Hsieh & Lee (2021), we study the behavior of the outputs in hidden layers for diverse networks to observe if differences arise when these networks are fed with either natural or adversarial examples.

However, here we will use the MNFs technique that notably improves the classical mean-field for BNNs utilized commonly in the literature. This way, we can significantly examine discrepancies when BNNs provide accurate uncertainty estimates. Furthermore, detecting adversarial examples is somehow associated with the model robustness, which in the case of BNNs, could suggest the best strategy for locating stochastic layers in the architecture. To do so, we design both deterministic and BNN networks, where the latter include MNF layers in either only the top of the model, called Bayesian Last Layer (BBL) or in the entire architecture (Full Bayesian), as we observe in Figure 1.

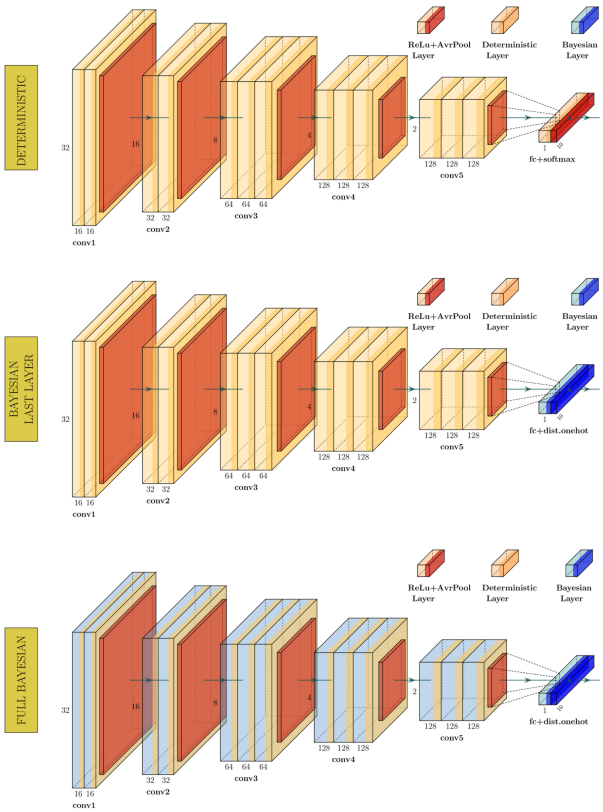

FIGURE 1: Illustration of the VGG16 network for the top: Deterministic, middle: Bayesian Last Layer, and bottom: Fully Bayesian. Yellow and blue color stand for deterministic and probabilistic layers respectively.

All models use VGG16 and VGG19 architectures (Simonyan & Zisserman, 2014), along with a one-hot categorical distribution at the end for BNNs, and the softmax activation for the deterministic counterpart. The models were trained using MNIST (Deng, 2012) and Fashion-MNIST (Xiao et al., 2017) 32×32 black and white images in ten different classes.

The dataset was split into $80\% - 20\%$ for training and testing, and the latter was split again into a holdout set of $80\% - 20\%$ for training and testing folds. The purpose of the training folds is to obtain a surrogate distribution based on the output layers for each category. On the other hand, the testing folds evaluate the robustness of the model. This is done by duplicating the test fold dataset, and one of these copies is attacked to observe how much its distribution changes concerning the surrogate distribution, and how close the non-attacked fold distribution is from the surrogate one. In addition, PCA is used to reduce the dimensionality of the output layers to 32. This PCA is fit with the training fold data and applied to all outputs. Comparisons among distributions are made through the Wasserstein and Energy distances. PGD and C&W attacks were implemented with stochastic optimization to provide stronger effects on the probabilistic layers, and two perturbation values (or learning rate values for C&W) were used in each scenario. Finally, we employ MNF layers[1][2] and Reparameterization trick[3] approach to observe the impact of flexible posterior distributions with respect to the standard mean-field normal.

5. Experiments & Results

Figure 2 displays the distribution in each model for one example from the dataset folds. Each column refers to the Deterministic network, Full Bayesian, and Bayesian Last Layer; concerning the level at which we extract the feature output to originate the distribution (last layer or convolutional blocks). ModelLogit and ConvBlock refer to extracting the output features from the last layer and the combination of only convolutional layers respectively. Notice that the adversarial provides wider distributions when BNNs are considered and the discrepancy concerning the surrogate distribution is more evident. The first result that we can report from the experiments is how BNNs outperform the deterministic models in detecting adversarial examples by reporting significant deviation from the surrogate distribution. Even if there exists a significant discrepancy between training and adversarial distributions, the test natural distribution does not follow completely the training behavior. This effect can be explained due to the stochastic nature of the model which was to attempt to approximate the training distribution with only a few samples. However, those samples taken by passing a few times test data into the model allow for following the main properties such as the multiples modes and asymmetries. These properties are better extracted with full BNNs, which is consistent since we recollect more samples from these architectures.

[1] https://github.com/janosh/tf-mnf
[2] https://github.com/akashrajkn/waffles-and-posteriors
[3] https://www.tensorflow.org/probability

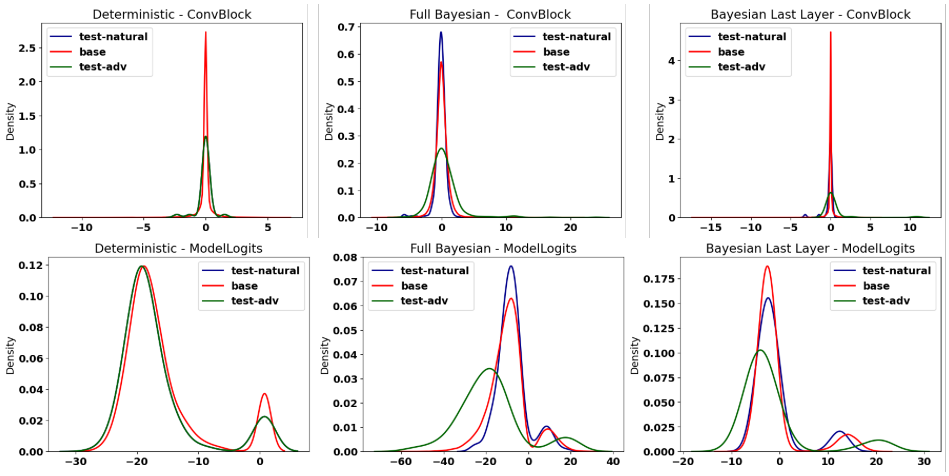

FIGURE 2: Distributions of hidden layer output of the adversarial BNN and deterministic detector. Each column stands for the deterministic approach, BLL, and Full Bayesian. The data chosen was MNIST for the three-category. The attack employed is PGD with $\epsilon = 0.1$.

Similarly, Figure 3 illustrates the behavior in the distributions of adversarial and natural images for the reparameterization trick. In the full BNN scenario, we can observe how adversarial distribution has a large amplitude in comparison with the other distributions. This behavior can be seen only under this configuration.

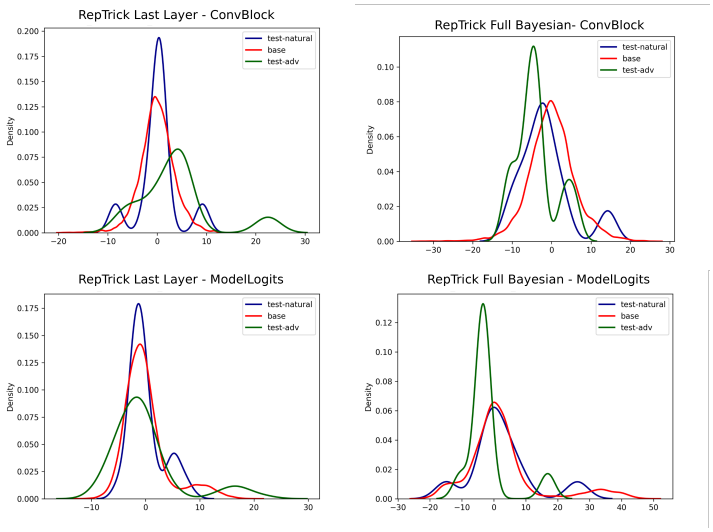

FIGURE 3: Distributions of hidden layer output of the adversarial BNN using the reparameterization trick approach. Each column stands for the BLL and Full Bayesian. The data chosen was MNIST for the three-category. The attack employed is PGD with $\epsilon = 0.1$

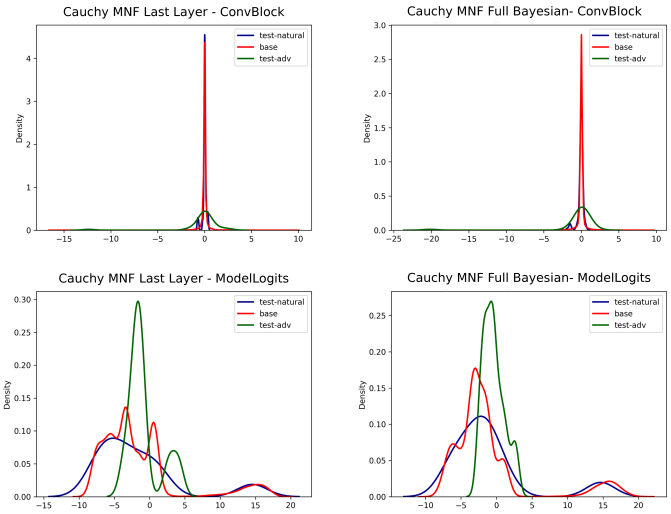

FIGURE 4: Distributions of the hidden layer from Cauchy MNF Bayesian networks. The data chosen was MNIST for the three-category. The attack employed is PGD with $\epsilon = 0.1$.

On the other hand, Tables 1-2 summarize the results obtained through experiments across MNIST and while Tables 3-4 display the results for Fashion-MNIST datasets along with both VGG16-19 architectures. Here, we report the distance value differences between natural and adversarial distributions. Larger distances imply better detection of adversarial examples. Let us start analyzing the effect of the amplitude of the attack on distances. For C&W attacks we can observe that distances are independent of the amplitude of the attack, while for PGD the distances become proportional with ϵ values. This can be explained since PGD turns out to be more aggressive than C&W.

TABLE 1: mnist-VGG16

MNIST-VGG16										
Layer	**Arch**	**CW**				**PGD**				**ECE**
		0,005		**0,01**		**0,15**		**0,3**		
		Ener	**Wass**	**Ener**	**Wass**	**Ener**	**Wass**	**Ener**	**Wass**	
Logits	**BLLMNF**	0,46	1,35	0,43	1,25	0,09	0,26	0,27	0,85	0,22
	BLLRe	0,01	0,03	0,00	0,01	0,20	0,73	0,00	0,01	0,25
	FullMNF	**0,68**	**1,93**	**0,54**	1,47	0,06	1,66	0,38	1,69	0,49
	FullRe	0,27	1,70	0,26	**1,65**	**0,65**	**2,09**	**1,30**	**3,86**	**0,19**
Conv	**BLLMNF**	0,26	0,57	0,25	0,55	0,07	0,13	0,10	0,24	0,22
	BLLRe	0,07	0,41	0,10	0,45	0,15	0,51	**0,30**	**1,20**	0,25
	FullMNF	0,08	0,14	0,06	0,11	0,06	0,12	0,08	0,20	0,49
	FullRe	**0,33**	**1,61**	**0,34**	**1,64**	**0,18**	**0,70**	0,24	0,85	**0,19**

Table 2: mnist-VGG19

MNIST-VGG19										
Layer	**Arch**	**CW**				**PGD**				**ECE**
		0,005		**0,01**		**0,15**		**0,3**		
		Ener	**Wass**	**Ener**	**Wass**	**Ener**	**Wass**	**Ener**	**Wass**	
Logits	**BLLMNF**	**0,76**	**2,44**	**0,70**	**2,26**	0,12	0,47	0,24	1,08	0,50
	BLLRe	0,18	0,51	0,13	0,38	0,21	0,88	0,39	1,66	0,83
	FullMNF	0,46	1,20	0,32	0,80	0,40	1,19	0,43	2,08	**0,41**
	FullRe	0,33	1,71	0,31	1,62	**0,81**	**2,28**	**1,64**	**4,70**	0,68
Conv	**BLLMNF**	0,24	0,49	**0,22**	0,44	0,06	0,12	0,08	0,16	0,50
	BLLRe	0,07	0,59	0,05	0,41	**0,18**	0,76	0,32	1,56	0,83
	FullMNF	0,05	0,07	0,03	0,05	0,05	0,08	0,07	0,17	**0,41**
	FullRe	**0,25**	**1,49**	**0,22**	**1,39**	0,16	**0,77**	**0,41**	**1,76**	0,68

Table 3: Fashion-mnist-VGG16

FASHION_MNIST-VGG16										
Layer	**Arch**	**CW**				**PGD**				**ECE**
		0,005		**0,01**		**0,15**		**0,3**		
		Ener	**Wass**	**Ener**	**Wass**	**Ener**	**Wass**	**Ener**	**Wass**	
Logits	**BLLMNF**	**0,43**	**1,11**	**0,37**	**0,94**	0,59	2,01	0,80	2,68	5,21
	BLLRe	0,10	0,21	0,08	0,17	0,27	0,99	0,43	1,66	**1,75**
	FullMNF	0,19	0,59	0,16	0,50	1,09	**5,24**	1,80	**9,24**	2,93
	FullRe	0,22	0,85	0,19	0,76	**1,64**	4,12	**2,28**	6,35	2,06
Conv	**BLLMNF**	0,10	0,24	0,11	0,25	0,37	1,05	0,43	1,30	5,21
	BLLRe	**0,23**	0,77	0,17	0,66	**0,72**	**2,99**	**1,14**	**5,12**	**1,75**
	FullMNF	0,01	0,03	0,02	0,04	0,20	0,50	0,36	1,02	2,93
	FullRe	0,17	**0,87**	**0,18**	**0,91**	0,49	1,95	0,95	4,67	2,06

Table 4: Fashion-mnist-VGG19

FASHION_MNIST-VGG19										
Layer	**Arch**	**CW**				**PGD**				**ECE**
		0,005		**0,01**		**0,15**		**0,3**		
		Ener	**Wass**	**Ener**	**Wass**	**Ener**	**Wass**	**Ener**	**Wass**	
Logits	**BLLMNF**	**0,52**	**1,34**	**0,45**	**1,15**	0,76	2,39	0,82	2,63	4,34
	BLLRe	0,09	0,20	0,08	0,17	0,37	1,43	0,49	1,85	2,30
	FullMNF	0,18	0,59	0,14	0,47	0,75	**3,17**	1,38	**6,70**	**0,45**
	FullRe	0,20	0,85	0,18	0,77	**2,69**	0,90	**3,49**	0,53	2,23
Conv	**BLLMNF**	0,16	0,34	0,14	0,28	0,29	0,87	0,32	0,90	4,34
	BLLRe	0,11	0,41	0,09	0,27	0,81	**3,24**	1,03	**4,47**	2,30
	FullMNF	0,02	0,03	0,01	0,03	0,12	0,25	0,20	0,50	**0,45**
	FullRe	**0,29**	**1,33**	**0,28**	**1,26**	**2,82**	2,93	**3,78**	2,02	2,23

This effect can be explained in the sense that all variability flows through the network and collocates at the top of the network. However, the foremost result arrives when we compute the difference between both adversarial and natural distances, where full Bayesian models report the highest differences. Therefore, designing BNNs will exhibit a trade-off between robustness and simplicity, i.e., Full BNNs can achieve good robustness for detecting anomalous data, but due to the duplication of weights concerning the deterministic counterpart, their training and inference phase becomes longer. Besides, this robustness is implicit not only when probabilistic layers cover the entire model, but also when architectures are bigger as we found when VGG16 and VGG19 are contrasted. Regarding the adversarial attacks, notice that PGD causes wide value concerning the C&W, which leads to the fact that the latter attack is stronger enough to deceive neural networks. Even so, BNNs have proven to be robust against its attacks. When analyzing the complexity of network architectures and the distances set up through the detection process, various outcomes are seen. It is crucial to take into account the block, the attack, and the applied ϵ value. Regarding the C&W attack, it has been observed that within the LogitsBlock, the BLLMNF architecture exhibits a tendency to amplify distances as the network complexity grows. The behavior in the BLLRe and Full architectures is the opposite. For the ConvBlock, the BLLRe tends to have lower distances with more complex architectures, which is contrary to the other three architectures where distances increase with the complexity of the architecture. For PGD, specifically in the ModelLogit, an increase in the ϵ value leads to a reduction in the distances between architectures. When the ϵ value is low, complex designs exhibit larger distances, but increasing the ϵ value leads to a decrease in the distances between architectures. For ConvBlock in PGD attacks, regardless of whether it is used in MNF networks with BLL or Full complexity, the distances drop as the architectures become more complicated and independent of the ϵ value used for the attack.

In the case of the ConvBlock within a BLLRe or FullRe, the behavior is analogous to that of the BLL ModelLogits, where an increase in the ϵ value leads to a decrease in distances. Concerning UCE, it is feasible to determine that the presence of good calibration does not necessarily indicate high detection. Certain networks may exhibit superior UCE in comparison to other networks, while also demonstrating improved distances. This is exemplified by the LogitsBlock of the BLLMNF VGG19 architecture in the MNIST dataset. Similarly, there may be situations when the structure of particular data shows the highest UCE, but its ability to detect is not ideal, as seen in the ConvBlock of the FullMNF VGG19 in Fashion-MNIST dataset.

5.1. Effect of Priors on Adversarial Detections

Table 5 displays the outcomes of the C&W experiment by comparing the same model architecture with two novel MNF methodologies: Gumbel (Gum) and Cauchy (Cau) priors, while Figure 4 displays several distributions derived from the Cauchy MNF version. Notice how the detection capability of the network is improved by changing the MNF prior to Gumbel, as compared to previous

cases of the ModelLogits in Full architecture. It is worth mentioning once more that Gumbel is not the model with the highest UCE among all the models in this block. Similarly, Cauchy produces superior detection outcomes within the same block but in the BLL architecture. The ConvBlock for BLL has selected MNF as a superior detector in this particular circumstance. Finally, inside the same block, the FullRe model has superior detection capacity.

Table 5: Priors

MNIST-VGG16						
Layer	Arch	CW				ECE
		0,005		0,01		
		Ener	Wass	Ener	Wass	
Logits	**BLLCMNF**	0,46	1,35	0,43	1,25	0,40
	BLLCRe	0,01	0,03	0,00	0,01	0,25
	BLLCMNF Cau	**0,97**	**2,82**	**0,92**	**2,67**	0,94
	BLLCMNF Gum	0,65	1,88	0,64	1,83	0,63
	FullMNF	0,68	1,93	0,54	1,47	0,40
	FullRe	0,27	1,70	0,26	1,65	**0,19**
	FullMNF Cau	0,63	1,87	0,62	1,82	0,31
	FullMNF Gum	0,69	2,14	0,68	2,09	0,46
Conv	**1CMNF**	0,26	0,57	0,25	0,55	0,40
	1CRe	0,07	0,41	0,10	0,45	0,25
	1CMNF Cau	0,24	0,56	0,22	0,53	0,94
	1CMNF Gum	0,19	0,55	0,18	0,52	0,63
	FullMNF	0,08	0,14	0,06	0,11	0,40
	FullRe	**0,33**	**1,61**	**0,34**	**1,64**	**0,19**
	FullMNF Cau	0,22	0,57	0,20	0,54	0,31
	FullMNF Gum	0,17	0,52	0,17	0,50	0,46

5.2. Discussion

The results demonstrate that hidden Bayesian layers can effectively detect adversarial examples, with prior selection playing a key role in improving detection performance. Calibrated neural networks provide reliable uncertainty estimates, where the distribution width accurately reflects adversarial detection. Ignoring this calibration can lead to increased false negatives, particularly in cases of overestimation. In standard applications, Bayesian Neural Networks (BNNs) are typically calibrated post-training, as calibration adjusts the distribution width while keeping the mean fixed, optimizing reliability metrics. However, in adversarial detection, well-calibrated uncertainties during training are crucial. Thus, methods that reduce variational inference (VI)-induced bias are particularly advantageous. Additionally, as discussed in Fortuin et al. (2021), prior selection depends on architecture and influences both model performance and cold posterior effects. Our findings align with this observation: (light) heavy-tailed priors enhance detection, likely because appropriate priors mitigate cold posterior effects and yield a more accurate Bayesian representation. Our code is available at ADVBNN.

5.3. Conclusion

In this paper, we found out that Bayesian neural networks outperform the deterministic models in detecting adversarial examples generated by several attacks. We also report a strong performance in detecting adversarial examples when Fully Bayesian networks are employed due to the potential of their hidden layers to provide distinct features (i.e., higher distances) between natural and adversarial examples. The use of MNFs in this work allows to the endowing of BNNs with flexible posterior distributions in such a way that uncertainty estimates are well-calibrated and therefore, our findings yield reasonable confidence levels. Also, we might argue that a well-adopted strategy for designing BNNs exhibits a trade-off between robustness and simplicity. Fully BNNs can achieve good robustness for detecting anomalous data, but their training and inference phases become longer with the duplication of weights for the deterministic counterpart. Furthermore, we explore the use of different priors in cases where MNF was applied. We have observed that the detection capability of the network improves when prior over the weights follows a Cauchy or Gumbel distribution. Nevertheless, these models were shown to be uncalibrated, which allows us to conclude no correlation between well-calibrated models and their robustness for detecting adversarial examples.

Concerning the findings derived from the experiments conducted in this study, the configuration of hybrid neural networks comprising both deterministic and Bayesian layers emerges as a promising avenue for the development of adversarial image detection methods. This approach considers the computational implications associated with fully Bayesian neural networks, as the incorporation of Bayesian structures generally entails a significant increase in resource consumption when compared to networks originally designed with deterministic architectures. Consequently, the implementation of a single Bayesian layer or a targeted subset of Bayesian layers could serve as a practical compromise, enabling the design of detection methods that are computationally efficient and potentially applicable in systems with limited processing capabilities. This stands in marked contrast to the approach adopted by Li, Tang, Hsieh & Lee (2021); Deng et al. (2021*b*); Huck Yang et al. (2022), in which the use of fully Bayesian neural networks—where all layers are of a Bayesian nature—plays a central role in the proposed methodology.

With regard to the development of Bayesian or hybrid neural network solutions, it is essential to further investigate their applicability using more complex resources, such as higher-resolution images and more sophisticated neural architectures compared to VGG16 and VGG19. The experiments conducted in this study serve as an entry point for future research efforts aimed at evaluating, under more demanding conditions, the computational feasibility of employing fully Bayesian or hybrid structures for image data comparable to CIFAR-10, or even for more industrial use cases involving real-time image capture by electronic devices, among other emerging technologies. Based on the experimental experience gained throughout this research, we observed the impact of such methods on both the training process and the generation of adversarial attacks across deterministic, Bayesian, and hybrid models. For this reason, we recommend that subsequent research efforts focus on increasing the complexity of the neural networks to be designed, as well as the datasets and types of attacks to be evaluated.

[Received: August 2024 — Accepted: May 2025]

References

Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Rajendra Acharya, U., Makarenkov, V. & Nahavandi, S. (2020), 'A Review of Uncertainty Quantification in Deep Learning: Techniques, Applications and Challenges', *arXiv e-prints* p. arXiv:2011.06225.

Andriushchenko, M., Croce, F., Flammarion, N. & Hein, M. (2019), 'Square Attack: a query-efficient black-box adversarial attack via random search', *arXiv e-prints* p. arXiv:1912.00049.

Ben Braiek, H., Reid, T. & Khomh, F. (2022), 'Physics-Guided Adversarial Machine Learning for Aircraft Systems Simulation', *arXiv e-prints* p. arXiv:2209.03431.

Bortsova, G., González-Gonzalo, C., Wetstein, S. C., Dubost, F., Katramados, I., Hogeweg, L., Liefers, B., van Ginneken, B., Pluim, J. P. W., Veta, M., Sánchez, C. I. & de Bruijne, M. (2020), 'Adversarial Attack Vulnerability of Medical Image Analysis Systems: Unexplored Factors', *arXiv e-prints* p. arXiv:2006.06356.

Carlini, N. & Wagner, D. (2016), 'Towards Evaluating the Robustness of Neural Networks', *arXiv e-prints* p. arXiv:1608.04644.

Cohen, G., Sapiro, G. & Giryes, R. (2019), 'Detecting Adversarial Samples Using Influence Functions and Nearest Neighbors', *arXiv e-prints* p. arXiv:1909.06872.

Craighero, F., Angaroni, F., Stella, F., Damiani, C., Antoniotti, M. & Graudenzi, A. (2021), 'Unity is strength: Improving the Detection of Adversarial Examples with Ensemble Approaches', *arXiv e-prints* p. arXiv:2111.12631.

Croce, F. & Hein, M. (2019), 'Minimally distorted Adversarial Examples with a Fast Adaptive Boundary Attack', *arXiv e-prints* p. arXiv:1907.02044.

Croce, F. & Hein, M. (2020), 'Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks', *arXiv e-prints* p. arXiv:2003.01690.

Deng, L. (2012), 'The mnist database of handwritten digit images for machine learning research [best of the web]', *IEEE Signal Processing Magazine* **29**(6), 141–142.

Deng, Z., Yang, X., Xu, S., Su, H. & Zhu, J. (2021*a*), 'LiBRe: A Practical Bayesian Approach to Adversarial Detection', *arXiv e-prints* p. arXiv:2103.14835.

Deng, Z., Yang, X., Xu, S., Su, H. & Zhu, J. (2021*b*), Libre: A practical bayesian approach to adversarial detection, *in* '2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)', IEEE Computer Society, Los Alamitos, CA, USA, pp. 972–982. https://doi.ieeecomputersociety.org/10.1109/CVPR46437.2021.00103

Dinh, L., Sohl-Dickstein, J. & Bengio, S. (2016), 'Density estimation using Real NVP', *arXiv e-prints* p. arXiv:1605.08803.

Fiedler, F. & Lucia, S. (2023), 'Improved uncertainty quantification for neural networks with Bayesian last layer', *arXiv e-prints* p. arXiv:2302.10975.

Fortuin, V., Garriga-Alonso, A., Ober, S. W., Wenzel, F., Rätsch, G., Turner, R. E., van der Wilk, M. & Aitchison, L. (2021), 'Bayesian Neural Network Priors Revisited', *arXiv e-prints* p. arXiv:2102.06571.

Gal, Y. (2016), Uncertainty in Deep Learning, PhD thesis, University of Cambridge.

García-Farieta, J. E., Hortúa, H. J. & Kitaura, F.-S. (2023), 'Bayesian deep learning for cosmic volumes with modified gravity', *arXiv e-prints* p. arXiv:2309.00612.

Goodfellow, I. J., Shlens, J. & Szegedy, C. (2014), 'Explaining and Harnessing Adversarial Examples', *arXiv e-prints* p. arXiv:1412.6572.

Graves, A. (2011), Practical variational inference for neural networks, *in* J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira & K. Q. Weinberger, eds, 'Advances in Neural Information Processing Systems', Vol. 24, Curran Associates, Inc., pp. 2348–2356.

Guesmi, A., Khasawneh, K. N., Abu-Ghazaleh, N. & Alouani, I. (2022), 'ROOM: Adversarial Machine Learning Attacks Under Real-Time Constraints', *arXiv e-prints* p. arXiv:2201.01621.

Henning, C., D'Angelo, F. & Grewe, B. F. (2021), 'Are Bayesian neural networks intrinsically good at out-of-distribution detection?', *arXiv e-prints* p. arXiv:2107.12248.

Huck Yang, C.-H., Ahmed, Z., Gu, Y., Szurley, J., Ren, R., Liu, L., Stolcke, A. & Bulyko, I. (2022), 'Mitigating Closed-model Adversarial Examples with Bayesian Neural Modeling for Enhanced End-to-End Speech Recognition', *arXiv e-prints* p. arXiv:2202.08532.

Hüllermeier, E. & Waegeman, W. (2019), 'Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods', *arXiv e-prints* p. arXiv:1910.09457.

Jimenez Rezende, D. & Mohamed, S. (2015), 'Variational Inference with Normalizing Flows', *arXiv e-prints* p. arXiv:1505.05770.

Laves, M.-H., Ihler, S., Kortmann, K.-P. & Ortmaier, T. (2019), 'Well-calibrated model uncertainty with temperature scaling for dropout variational inference'.

Lee, K., Lee, K., Lee, H. & Shin, J. (2018), 'A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks', *arXiv e-prints* p. arXiv:1807.03888.

Li, Y., Cheng, M., Hsieh, C.-J. & Lee, T. C. M. (2021), 'A Review of Adversarial Attack and Defense for Classification Methods', *arXiv e-prints* p. arXiv:2111.09961.

Li, Y., Tang, T., Hsieh, C.-J. & Lee, T. C. M. (2021), 'Detecting Adversarial Examples with Bayesian Neural Network', *arXiv e-prints* p. arXiv:2105.08620.

Louizos, C. & Welling, M. (2017), 'Multiplicative Normalizing Flows for Variational Bayesian Neural Networks', *arXiv e-prints* p. arXiv:1703.01961.

Lu, J., Issaranon, T. & Forsyth, D. (2017), 'SafetyNet: Detecting and Rejecting Adversarial Examples Robustly', *arXiv e-prints* p. arXiv:1704.00103.

Ma, X., Li, B., Wang, Y., Erfani, S. M., Wijewickrema, S., Schoenebeck, G., Song, D., Houle, M. E. & Bailey, J. (2018), 'Characterizing Adversarial Subspaces Using Local Intrinsic Dimensionality', *arXiv e-prints* p. arXiv:1801.02613.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D. & Vladu, A. (2017), 'Towards Deep Learning Models Resistant to Adversarial Attacks', *arXiv e-prints* p. arXiv:1706.06083.

Moosavi-Dezfooli, S.-M., Fawzi, A. & Frossard, P. (2015), 'DeepFool: a simple and accurate method to fool deep neural networks', *arXiv e-prints* p. arXiv:1511.04599.

Mukherjee, S., Sen, B. & Sen, S. (2023), 'A Mean Field Approach to Empirical Bayes Estimation in High-dimensional Linear Regression', *arXiv e-prints* p. arXiv:2309.16843.

Simonyan, K. & Zisserman, A. (2014), 'Very Deep Convolutional Networks for Large-Scale Image Recognition', *arXiv e-prints* p. arXiv:1409.1556.

Smith, L. & Gal, Y. (2018), 'Understanding Measures of Uncertainty for Adversarial Example Detection', *arXiv e-prints* p. arXiv:1803.08533.

Wang, Y., Sun, T., Li, S., Yuan, X., Ni, W., Hossain, E. & Poor, H. V. (2023), 'Adversarial Attacks and Defenses in Machine Learning-Powered Networks: A Contemporary Survey', *arXiv e-prints* p. arXiv:2303.06302.

Watson, J., Andreas Lin, J., Klink, P., Pajarinen, J. & Peters, J. (2021), Latent derivative bayesian last layer networks, *in* A. Banerjee & K. Fukumizu, eds, 'Proceedings of The 24th International Conference on Artificial Intelligence and Statistics', Vol. 130 of *Proceedings of Machine Learning Research*, PMLR, pp. 1198–1206. https://proceedings.mlr.press/v130/watson21a.html

Wu, H., Yunas, S., Rowlands, S., Ruan, W. & Wahlstrom, J. (2021), 'Adversarial Driving: Attacking End-to-End Autonomous Driving', *arXiv e-prints* p. arXiv:2103.09151.

Xiao, H., Rasul, K. & Vollgraf, R. (2017), 'Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms', *arXiv e-prints* p. arXiv:1708.07747.

Xie, Z., Brophy, J., Noack, A., You, W., Asthana, K., Perkins, C., Reis, S., Singh, S. & Lowd, D. (2022), 'Identifying Adversarial Attacks on Text Classifiers', *arXiv e-prints* p. arXiv:2201.08555.

Zhang, C., Butepage, J., Kjellstrom, H. & Mandt, S. (2017), 'Advances in Variational Inference', *arXiv e-prints* p. arXiv:1711.05597.