Adversarial Robustness of Uncertainty Aware Deep Neural Networks

Ali Harakeh
ali.harakeh@utoronot.ca
Institute for Aerospace Studies
North York, Ontario

1 INTRODUCTION

Recent progress in deep neural network literature paved the way for applications in many safety critical domains such as autonomous vehicles, human surveillance, and medical diagnosis. Unfortunately, deep neural networks can be very vulnerable to maliciously generated adversarial examples. Such adversarial examples are usually generated by perturbations that are imperceptible to humans. This project proposes to study the ability of neural networks to become aware of such perturbations through estimating the uncertainty in their predictions.

Two sources of uncertainty can be identified in any machine learned model. *Epistemic* or model uncertainty is the uncertainty in the model's parameters, usually as a result of the confusion about which model generated the training data, and can be explained away given enough representative training data points [4]. On the other hand, *aleatoric* or observation uncertainty results from the stochastic nature of the observed input, and persist in network output despite expanded training on additional data [7].

Methods to estimate both uncertainty types in deep neural network models have been recently proposed [7], with applications in one-to-one perception tasks such as semantic segmentation and monocular depth regression. Recent work [5] argued that idealized Bayesian neural networks (ones that estimate epistemic uncertainty) are not susceptible to adversarial attacks, but was rejected by the reviewers in ICLR 2019, due to the fact that the 'ideal' settings in the paper can never be achieved in reality. However, no previous work has studied the ability of aleatoric uncertainty estimation mechanisms to reject adversarial examples.

In summary, I aim to answer the following question: **Can a neural network mitigate the effects of adversarial attacks by estimating the uncertainty in its predictions?** To do so, the following report is comprised of the following sections. Section 2 presents an overview of recent uncertainty estimation mechanisms in deep neural networks and prominent adversarial attacks used to develop the experiments. Section 4 presents a description of the methodology used for experiments, the experimental results, and a brief discussion on these results. Section 5 concludes the report with the most prominent lessons learnt from the performed experiments. Finally Appendix A shows how much I was able to achieve given what was planned for the project.

2 LITERATURE REVIEW

Uncertainty Estimation In Deep Neural Networks:

Epistemic (Model) Uncertainty: To capture the epistemic uncertainty of a deep neural network model, a prior distribution is imposed over its parameters θ to compute a posterior distribution $p(\theta|\mathcal{D})$ over the set of all possible parameters given the training data $\mathcal{D}$. A marginal distribution is then computed for every input image x_i as:

$$p(\hat{y}_i|x_i, \mathcal{D}) = \int_{\theta} p(\hat{y}_i|\mathbf{x}_i, \mathcal{D}, \theta)p(\theta|\mathcal{D})d\theta, \tag{1}$$

where $\hat{y}_i$ is the classification logits of the neural network. Unfortunately, the integral is usually intractable, and approximate inference is usually needed to evaluate the posterior distribution. A simple and computationally efficient Monte-Carlo sampling method, Monte-Carlo Dropout [4], allows drawing **i.i.d** samples from Eq. (1) by performing neural network inference with dropout enabled. The output of the neural network can then be expressed as:

$$\hat{y}_i = \frac{1}{T}\sum_{t=1}^{T} softmax(f(\mathbf{x}_i, \theta_t)), \tag{2}$$

where T is the number of runs performed with dropout, and $f(\mathbf{x}_i, \theta_t)$ is the logit output from the neural network. It has to be noted that one can also use more computationally expensive and data hungry ensemble methods to approximate the integral.

Aleatoric (Process) Uncertainty: Motivated by heteroscedastic regression[9], a loss attenuation formulation has been proposed [7] that estimates the variance of the classification output of a deep neural network model by modifying the training loss as follows:

$$\begin{aligned} L &= \frac{1}{N}\sum_{i=1}^{N} H(y_i, \hat{y}_i) \\ \hat{y}_i &= softmax(f(x_i, \theta) + \sigma(x_i, \theta)\mathcal{N}(0, I)), \end{aligned} \tag{3}$$

where I is the identity matrix, and $\sigma(x_i, \theta)$ is the estimated heteroscedastic aleatoric variance. I will be using these two

formulations in the experiments presented bellow to estimate both types of uncertainty.

Adversarial Examples

Szegedy et al. [13] discovered that despite deep neural networks having an extraordinary accuracy when applied to the classification task, a small imperceptible perturbations to the image can cause a catastrophic misclassification of previously well classified examples. Such perturbed examples are usually maliciously generated and are referred to as Adversarial Examples. This section provides an overview of the four types of attack generators used in the experiments.

Fast Gradient Sign Method (FGSM) [6]: FGSM is a method to solve the optimization problem:

$$\rho = \epsilon sign(\nabla J(\theta, x, \ell)), \tag{4}$$

where ρ is the required adversarial perturbation, ϵ is a small value that restricts the norm of the perturbation, and $\nabla J(...)$ computes the gradient of the cost function around the current value of the model parameters θ with respect to the input x.

Jacobian-Based Saliency Map Attack (JSMA) [12]: JSMA restricts the l_0 norm of the perturbation instead of the regularly used l_∞ or l_2 norm. The algorithm thus modifies only a few pixels in the image instead of perturbing the whole image to fool the classifier. This is performed by modifying the pixels of the clean image one at a time, while monitoring the effects of the change on the resulting classification output through the a gradient based saliency map. A larger value in this map indicates a higher probability of fooling the classifier by pushing its classification output towards a target class. Once this process is done, the algorithm generates the adversarial example by choosing the k most effective pixels and altering them, where k is restricted by the l_0 norm.

Carlini and Wagner Attacks (CW) [2]: CW attacks can use l_0, l_2 and l_∞ norms to produce adversarial examples that are much more powerful than previous approaches, while keeping the perturbations quasi-imperceptible. The reader is referred to [2] for the exact optimization procedure as it is too long and complex to be described here.

Black Box Attack (BB) [11]: Unlike all previously discussed attacks, BB uses only the input-output relation of a classifier to construct adversarial examples, effectively treating the classifier as a black-box. Heuristics are designed to query a black box classifier a limited number of times. The attack strategy is to train a substitute network on a small number of initial queries, then iteratively perturb inputs based on the substitute network's gradient information to augment the training set. The Substitute DNN training algorithm is:

(1) Acquire a small initial dataset.
(2) Select an architecture for the substitute network thatâĂŹs sensible for the domain.
(3) Query the oracle (target) network for labels to any unlabeled datapoints.
(4) Train the substitute network on current datapoints.
(5) Augment the dataset with Jacobian based data augmentation (JBDA) perturbations.
(6) Repeat as necessary.

Once the substitute network is trained, one can use standard adversarial attack algorithms to generate adversarial examples.

3 METHODOLOGY

To test whether a neural network gains adversarial robustness by estimating the prediction uncertainty, the following experiments are performed. The neural network is first modified to estimate epistemic or aleatoric uncertainty. The neural network is then attacked by various adversarial perturbations, generated from the algorithms described in Section 2. The effectiveness of these attacks are then measured on the vanilla unmodified network, the network with epistemic uncertainty estimation, and the network with aleatoric uncertainty estimation.

Neural Networks Models and Datasets

Two classification datasets are used for testing, the MNIST handwritten digits dataset [3], and the CIFAR-10 natural image dataset [8]. Fig. 1 shows the neural network architectures used on each of the two datasets, as well as example images from both datasets. On MNIST, a simple network is trained, comprising of three $3x3$ convolution layers each with 64 filters followed by a Fully Connected layer to output 10 logit values corresponding to the 10 digits in the dataset. On CIFAR-10, a more complex fully convolutional network is trained, comprising of two $3x3$ convolution layers each with 64 filters, followed by a $2x2$ average pooling layers, followed by another $3x3$ convolution layer with 10 filters, and finally one average pooling layer over the full feature map to provide the logits. Those two architectures are the ones used usually to test adversarial attacks and as such has been chosen for the experiments.

Uncertainty Estimation Algorithms

To estimate epistemic uncertainty, integration over the parameters of the neural network is approximated using MC-Dropout [4]. The two architectures are modified by adding dropout layers with 0.5 dropout probability after the convolutional layers as shown in Fig. 1, which are kept to run

Type	Algorithm	Description
Neural Networks	Basic CNN	A basic neural network comprising of 3 convolutional layers and 1 fully connected layer.
	Fully Convolutional Network	A moderately complex neural network comprising of convolutional and average pooling layers
Uncertainty Estimation Methods	Monte-Carlo Dropout[7]	An approximation to the integral in equation (1), used for epistemic uncertainty estimation.
	Heteroscedastic Regression [7]	Allows nueral network to output a variance that is used as described in equation (3).
Adversarial Perturbation Generators	FGSM [6]	A fast method used to optimize the image pixels to increase a target loss according to equation (??).
	CW [12]	Uses the L_2 norm to generate adversarial perturbations. Stronger than FGSM.
	JSMA [12]	Uses the L_0 norm to optimize for the minimum number of non-zero pixels that can be use to fool an neural network.
	BB [11]	Uses a surrogate neural network to generate adversarial perturbations. Does not require access to neural network parameters.

Table 1: A summary of the algorithms provided in Section 3. For both neural networks, both uncertainty estimation mechanisms were incorporated into the implementation. For Basic CNN, all adversarial attack algorithms were used. For Fully Convolutional Network, only FGSM was used due to time restrictions.

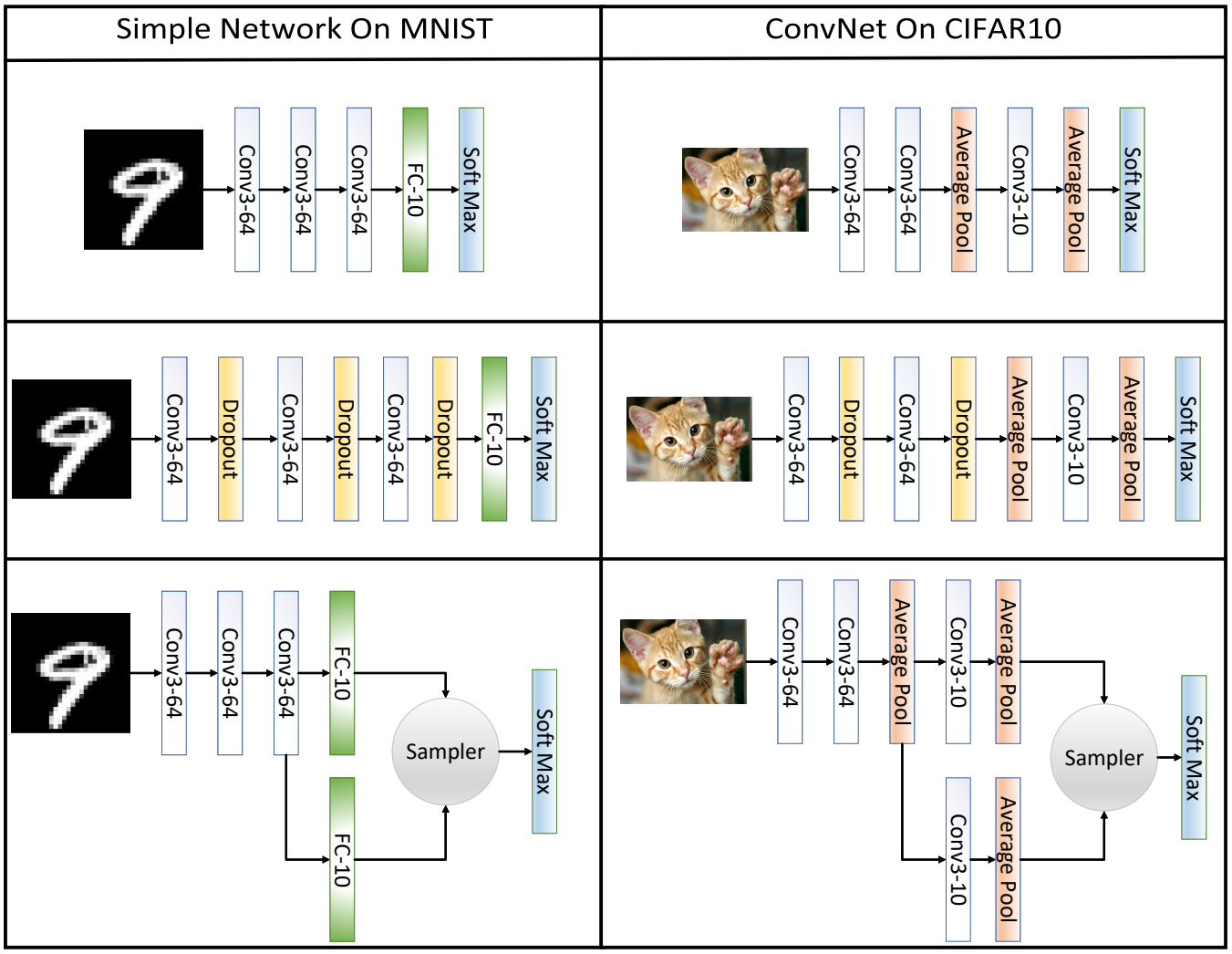

Figure 1: Neural Network architectures used on both datasets for experiments. *Top Row:* vanilla architectures. *Middle Row:* Modified Bayesian architectures to integrate over the neural network parameters. *Bottom Row:* Modified architectures used to estimate the variance and sample from a Gaussian distribution describing the logits.

during inference. The results of T runs are then averaged according to equation (2) to get the mean, which can be then computed to get the variance as a measure of uncertainty.

To estimate aleatoric uncertainty, I follow the method of [7]. A Gaussian distribution is put over the logits output of the neural network. The variance of this distribution is estimated by adding and additional layer as shown in Fig. 1. Logits are then sampled from the Gaussian distribution, averaged, and passed through the softmax to get the final classification output.

Generating Adversarial Perturbations:

Adversarial perturbations are generated using FGSM [6], JSMA [12], CW [1], and BB [11], all of which are described in Section 2. FGSM and BB is used to generate a perturbation for every test image in the dataset, allowing for the computation of adversarial accuracy on the whole dataset. On the other hand, JSMA and CW are targeted attacks, and as such, 10 repeated runs of 90-100 adversarial examples are generated for each of the attacks. The minimum and maximum fooling rate of these attacks is used as the measure of performance of the neural network.

Implementation Details

The code to perform these experiments is provided as an attachment to this report. Datasets are automatically downloaded as needed by running the specific experiment code. Neural networks and uncertainty estimation algorithms are implemented using TensorFlow, a deep learning API. To generate adversarial examples, the Cleverhans [10] library is used which works natively with TensorFlow based models. All hyperparameters and training details are chosen based on recommended values from Cleverhans tutorials, and can be found in the code submitted alongside this report. It has to be noted that the code provided is a modified version of Cleverhans tutorials, where I added custom neural networks that employ the described uncertainty estimation mechanisms and made the command prompt interface easier to use by non-experts.

4 RESULTS AND DISCUSSION

Quantitative Results

Table 2 provides quantitative results for all experiments performed on both datasets. The three variants of the neural network architectures shown in Fig. 1 are trained and evaluated for each of the four adversarial attack methods. This is indicated in the column header "Defense Type" to indicate uncertainty mechanisms used to alleviate the effect of an adversarial attack.

I will begin the analysis with the MNIST dataset. First it has to be noted that the accuracy of basic CNN on MNIST flucuates around 99%. FGSM is shown to reduce the accuracy of basic CNN by an order of magnitude, from 99.37% to 9.96%. When epistemic uncertainty is estimated, the accuracy on FGSM generated adversarial examples increases to 22.92%. Aleatoric uncertainty estimation does not seem to help at all in the case of FGSM, and the accuracy on adversarial examples remain low at 8.75%. Carlini and Wagner attacks are extremely powerful when used, resulting in a fooling rate of around $99 - 100\%$. That means that out of 100 adversarial perturbed images, at least 99 successfully trick the neural network to provide erroneous classification. Surprisingly, epistemic uncertainty estimation for basic CNN reduces the fooling rate to a maximum of 37%. Although to a lower degree, aleatoric uncertainty estimation for basic CNN also reduces the fooling rate to a maximum of 80%. A similar trend to CW attacks can be observed in JSMA attacks, where the original fooling rate of $89 - 92\%$ was reduced to a maximum of 27% with epistemic uncertainty estimation, and a maximum of 81% with aleatoric uncertainty estimation. Finally, the black box attack resulted in an adversarial accuracy of around 67.78%. Surprisingly, neither epistemic nor aleatoric uncertainty estimation mechanisms worked to increase the accuracy, which remained around $62 - 63\%$.

A final experiment that needs to be discussed uses a more complex Fully Convolutional Neural Network trained on the CIFAR-10 dataset. For this experiment, FGSM adversarial perturbations are used to attack the neural network, resulting in a decrease in accuracy from 77.84% to 9.98%. Surprisingly, the improvement in accuracy from epistemic uncertainty estimation is much lower than the improvement observed using the same attack type on MNIST resulting in an accuracy of 12.44%. Similar to the neural network on MNIST, aleatoric uncertainty failed to help increase the accuracy on adversarial examples generated by FGSM.

Observations and Discussion

This section will provide a description of deductions that can be derived from the above experiments in bullet point format.

Adversarial examples are out of distribution examples, created through the exploitation of point estimates of neural network parameters. This phenomenon has been previously discussed in [5], where the claim was that truly Bayesian neural networks cannot be fooled by adversarial examples. A confirmation of this phenomenon in my experiments is shown by observing that the estimation of epsitemic uncertainty improves the neural network performance in most adversarial settings. The estimation of epistemic uncertainty requires the marginalization over the neural network weights, that is taking every set of possible weights into consideration, which is usually approximated through monte-carlo dropout.

Another confirmation for the above hypothesis, is the inability of neural networks used in my experiments to identify

Dataset	Network	Attack Type	Defense Type	Accuracy (%)	Adversarial Accuracy (%)	Fooling Rate (%)
MNIST	**Basic CNN**	**FGSM**[6]	None	99.37	9.96	-
			Epistemic	98.50	22.92	-
			Aleatoric	99.35	8.75	-
		CW[1]	None	99.30	-	99-100
			Epistemic	98.37	-	30-37
			Aleatoric	99.32	-	67-80
		JSMA[12]	None	99.35	-	89-92
			Epistemic	98.62	-	22-27
			Aleatoric	99.34	-	73-81
		BB[11]	None	99.32	67.78	-
			Epistemic	98.51	63.29	-
			Aleatoric	99.20	62.08	-
CIFAR10	**Fully Convolutional Network**	**FGSM**[6]	None	77.84	9.98	-
			Epistemic	76.28	12.44	-
			Aleatoric	78.00	10.38	-

Table 2: Results of various adversarial attacks on two datasets using two neural network architectures. Accuracy measures the percent correct classification, the higher the better. Fooling rate measures how likely the architecture is to be fooled by an attack, the lower the better. On CIFAR-10, only FGSM was tested due to lack of time.

adversarial perturbation as input noise. Given the intuition that perturbation is added noise, I was hoping that perturbed examples will have a higher aleatoric variance output for the logits. However, the results show that such examples are actually classified with low variance as the wrong class.

The blackbox attack of [11] work even when epistemic uncertainty is estimated by the neural network. The mode of action of the blackbox attack proposed by [11] proposed to produce adversarial examples on a **surrogate** system to that that is observed as a blackbox. Epistemic uncertainty estimation cannot help in such cases, as the attack algorithm does not care about the black box weights! This phenomenon can be observed by noticing that the blackbox adversarial perturbations were able to maintain the same adversarial accuracy in the neural network even after incorporating epistemic uncertainty estimation.

Monte-Carlo dropout might not be a good approximation of marginalization over the weights. Monte-Carlo dropout is used to approximate marginalization over weights, and hence estimate epistemic uncertainty, according to equation (2). However, it can be noted that the effect of such approximation varies between neural network architectures. The improvement in accuracy on MNIST using basic ConvNet is much better than on CIFAR-10 using Fully Convolutional Network. I suspect that this is due to MC-Dropout failing to properly approximate the integral in more complex neural networks, especially ones with non-standard layers such as average pooling for example. However, the reason behind this phenomenon might also be due to the difference in datasets. CIFAR-10 is much more complex, and might allow more difficult adversaries, which would counteract the effect of epistemic uncertainty estimation.

Can a neural network mitigate the effects of adversarial attacks by estimating the uncertainty in its predictions?

When it comes to aleatoric uncertainty estimation, the answer is no. Both neural networks seem to no be able to factor out adversarial perturbations as input noise. However, when it comes to epistemic uncertainty estimation, the results are inconclusive. On MNIST, the results show quite a lot of improvement in adversarial robustness for all adversarial attack methods except the black box attack. On CIFAR-10 however, the results were not as good, with a mere 3% increase in accuracy for the neural network employing epistemic uncertainty estimation. In summary, it is recommended to not assume that uncertainty estimation in neural networks increase adversarial robustness.

Threats to Validity:

Before concluding, some threats to validity need to be noted. First, due to restrictions on computational power, relatively shallow neural networks were tested. The conclusions made might not transfer well to neural networks with tens of layers comprising the state-of-the-art. Second, most of the work was performed on the MNIST dataset, a very common approach in literature. As I discussed earlier, MNIST properties may not hold on any other datasets. This phenomenon has

also been observed and warned against in recent literature [1].

5 CONCLUSION

This report studies the adversarial robustness of neural networks given the estimation of epistemic and aleatoric uncertainty in their output predictions. The results show that at their current state, both epistemic and aleatoric uncertainty estimation mechanisms do not guarantee adversarial robustness. However, there is some evidence that a proper Bayesian neural network, with exact epistemic uncertainty estimates, might be immune to adversarial examples. Until an exact solutions for the marginalization in equation (1) is found, I do not recommend approximations such as Monte-Carlo dropout as a trusted adversarial defence mechanisms. As a final thought, methods to prove adversarial robustness are still the best approach to guarantee the safety of deep neural networks when it comes to malicious adversarial attacks.

REFERENCES

[1] Nicholas Carlini and David Wagner. 2017. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, 3–14.

[2] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 39–57.

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

[4] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning (ICML)*.

[5] Yarin Gal and Lewis Smith. 2018. Sufficient Conditions for Robustness to Adversarial Examples: a Theoretical and Empirical Study with Bayesian Neural Networks. (2018).

[6] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[7] Alex Kendall and Yarin Gal. 2017. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.).

[8] Alex Krizhevsky and Geoffrey Hinton. 2009. *Learning multiple layers of features from tiny images*. Technical Report. Citeseer.

[9] David A Nix and Andreas S Weigend. 1994. Estimating the mean and variance of the target probability distribution. In *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference On*.

[10] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, Alexander Matyasko, Vahid Behzadan, Karen Hambardzumyan, Zhishuai Zhang, Yi-Lin Juang, Zhi Li, Ryan Sheatsley, Abhibhav Garg, Jonathan Uesato, Willi Gierke, Yinpeng Dong, David Berthelot, Paul Hendricks, Jonas Rauber, and Rujun Long. 2018. Technical Report on the CleverHans v2.1.0 Adversarial Examples Library. *arXiv preprint arXiv:1610.00768* (2018).

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

[12] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 372–387.

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

A ACHIEVED REQUIREMENTS FOR PROJECT:

Bellow are bullet points to reflect what was achieved in this project compared to my goals. Achieved bullet points are in blue, non-achieved bullet points are in red, deprecated bullet points are in green. Deprecated bullet points are ones that were skipped due to their simplicity.

- Implement a fully connected Feedforward classification Neural Network on the MNIST dataset. (Instead, a moderate complexity convolutional neural network was used on MNIST).
- Use Cleverhans [10] to generate adversarial attacks using common algorithms such as the FGSM [6], Carlini and Wagner L2 [2] attacks for the given network.
- Implement Monte-Carlo dropout to extract model uncertainty [4, 7].
- Test if extracted uncertainty can be used to reject adversarial examples.

Hopefully Achievable Outcomes From the project:

- Implement ensembles to extract model uncertainty of given network.
- Implement heteroscedastic regression [4, 7] to extract input/process uncertainty for given network.

Highly Optimistic Outcomes Of the project, probably not achievable in 1 month:

- Extend the above experiments to moderately complicated convolutional neural networks.
- Extend the above experiments to CIFAR-10 dataset. (Managed to do this with one type of adversarial attacks, FGSM.)
- Extend the above methodology to complicated multi-task networks, specifically those used for object detection networks in 2D and 3D.