Adversarial Image Classification using Deep Neural Rejection

Janith Thanthulage; B523685
MEng Electronic and Electrical Engineering

Abstract: The use of deep neural networks has seen an increase in popularity in multiple different applications. Due to their impressive demonstratable performance across many disciplines, the reliance on such systems to add a level of automation have similarly increased. Consequently, the need for such systems to be robust to potential adversarial attacks at causing intentional misclassification becomes vital. In this work, we investigate the effectiveness of a newly proposed deep neural rejection mechanism against adversarial examples. We compare the effectiveness of this method to a previously proposed single layer neural rejection method and the arguably more common method of adversarial training. We perform a defence-aware evasion-type attack with the intent of being a 'worst case' attack on a defence system. We, to an extent, are able to confirm the effectiveness of deep neural rejection over other methods, and also offer some insight into the difference in the practicality of implementing the different methods of attack mitigation.

1. INTRODUCTION

Recent years have seen the increased use of computer vision in a multitude of different disciplines and contexts. More specifically, systems have been implemented with the intention to leverage the autonomy and accuracy that can be achieved by image recognition systems for complex tasks related to, but not limited to, medical diagnosis, security and autonomous driving [1-3]. A paradigm shift in the car manufacturing world can be witnessed as a growing number of manufacturers utilise image recognition and classification systems to allow for features such as automated parking, driving and alerting drivers of possible collisions on the road [1]. Additionally, computer vision systems have become an agent in the medical field to aid with biomedical research such as its use in recognising the prevalence of heart disease in samples instantaneously [2].

As we have become more dependent on such systems, the ability to robustly withstand possible attacks to the integrity of such systems goes from being an afterthought to a necessary point of focus. [4] first generated small perturbations on images used in an image classification task and was able to demonstrate vulnerabilities in classification systems by fooling such neural networks with high probability. These samples capable of causing such a misclassification were named as Adversarial Examples.

Consequently, this point of attention has in more recent years gathered an increasing amount of interest in both research and industry. The multifaceted task of testing whether computer vision models are robust remains a challenge, arguably due to the fact that measuring robustness cannot be given by a binary answer. As a result, many network architectures currently in use may show vulnerabilities due to both improper training combined with a lack of proper testing to measure the specific robustness to targeted attacks.

1.1 ADVERSARIAL ATTACKS

An adversarial attack is a deliberate attempt at deceiving a machine learning algorithm such as a Convolutional Neural Network (CNN) or Deep Neural Network (DNN). This is implemented through the use of malicious adversarial examples that are carefully crafted by an attacker to force the misclassification of a model to a particularly chosen input or through the addition of purposeful noise perturbations added to an input to mask relevant features and highlight others (again causing misclassifications). Note that adversarial samples are created at test time, after the DNN has already been trained by the defender, so do not perform any modification to the training process.

Impressive research has been conducted in performing attacks to cause misclassifications in the contexts detailed. One such work was the creation of perturbations to mimic graffiti that may exist on traffic signs and "thus hide in the human psyche"[5]. By creating such perturbations, it was found possible to misclassify objects such as a Stop sign to a Speed Limit 45 sign. Given the context of autonomous vehicular navigation, the existing vulnerability to attacks can be particularly dangerous.

Most research conducted has focused largely on the space of image classification systems due to perhaps the easiness of visualising the perturbations. The research conducted in [6] demonstrates the vulnerability of neural networks in a much more general application. The work conducted, demonstrated the ability to "construct targeted audio adversarial examples on automatic speech recognitions" systems: more specifically it demonstrated the ability to cause misclassifications on Mozilla's DeepSpeech model. This emphasises the existing vulnerability of neural networks in a more general application which may otherwise have been overlooked, further prompting the need for a methodology of defence against adversarial examples.

The threat of adversarial attacks have become an ever-increasing threat to autonomous systems dependant on trained machine learning networks. Despite being trained in secure environments and the networks themselves being contained privately; models still show susceptibility to adversarial perturbation.

1.2 AIM

This work aims to build upon and compare the effectiveness of existing methods of making a classification network robust.

Existing methods of defence against adversarial attacks can be generally split into two broad categories. One methodology is based around the robust optimization of a model, where a defender will explicitly incorporate knowledge of potential attacks into a model's training stage in addition to its usual training and testing. Adversarial training is a common utilisation of this methodology as to make an existing model more robust [7][8] . It has been shown by [9] that the retraining of a compromised network using its adversarial examples can have the benefit of providing an additional regularization beyond that provided by using methods such as dropout.

The other methodology is based around the idea of developing an architecture that is able to withstand or perform a level of rejection against adversarial examples through the identification of possible outlying perturbations. Such methods often utilise a form of model dimension reduction or model gradient smoothing [10-12]. A recent method proposed has been that of Deep Neural Rejection (DNR)[13], an extension of a previously proposed Neural Rejection (NR) method by [14], which both make use of RBF Support Vector Machines (SVM) trained on the higher level features of the compromised network with the objective of withstanding or alternatively rejecting adversarial images.

The effectiveness of these methods will be tested specifically against a projected-gradient-defence (PGD) type evasion attack of varying severities, see Section 2.5. The severity or strength of the attack (ε) will demonstrate the Euclidean distance (or L2 norm) of the adversarial image from the original: the greater the distance, the greater the perturbation and therefore the larger the expected possibility for misclassification. Hence, a measure of the robustness of a network or defence of a network can then be attributed to the attack strength required to cause its misclassification.

2. METHODOLOGIES

2.1 DATASET USED FOR IMPLEMENTATION

To demonstrate the effectiveness of attacks and defences, the MNIST database was used. It is a combination of two of NIST's (National Institute of Standards and Technology) databases, Special Database 1 and Special Database 3, which consists of handwritten digits written by high school students and employees of the United States Census Bureau respectively. It was predominately chosen as it is a pre-existing and well known dataset which is commonly used for testing classification systems and methodology. Use of this database offers the added ability to easily visualise the results of adversarial attacks and directly compare against the results of other published research which use the same database.

The MNIST dataset consists of 60,000 training and 10,000 testing grayscale images of size 28x28. Figure 1 illustrates nine example images from this dataset. All images were normalised to a range between 0 and 1 before the training and testing stages by dividing the original pixel values of the images by 255. As detailed in [13], the base DNN model and various defence architectures were similarly trained on a subset of 30,000 training images from the overall MNIST dataset.

The adversarial attack, discussed in Section 2.5.1, was performed on a random selection of 1000 images for each ε tested as to create a dataset of this size consisting of adversarial images. The various models were then tested against this dataset to determine the accuracy and rejection rate of each of the models. This was implemented 5 times and the results averaged and illustrated in Figure 11.

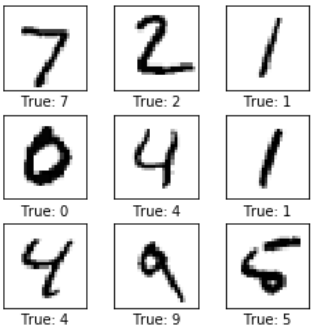

Figure 1 – Typical example images from the MNIST dataset

2.2 SOFTWARE (AND HARDWARE) USED FOR IMPLEMENTATION

The models and experiments tested were built using Python. The Keras and Scikit-learn libraries were predominantly used to build and train the DNN and SVMs respectively. The frameworks provided by these libraries are very well optimised and offer an attractive 'ease of use' element in addition to its large active community which can offer some additional support. Moreover, the Keras library offers compatibility with TensorFlow-GPU which allow for GPU computation, thus improving its speed of compilation greatly. This is particularly important due to the size of the datasets and features that are being trained.

2.3 DEEP NEURAL NETWORK CLASSIFIER

Before addressing the methods in which the model was attacked, we can first address the base classifier network. First proposed in [17], this architecture was also used as a control in the DNR [13] and NR [14] defence methods. The use of this base classifier architecture allows for a direct comparison between our findings and the results reported for each respective defence method. The details of each layer of the network is detailed in Table 1.

Operational Layer			Number of Filters	Size of each filter	Stride value	Padding value	Size of output
Input Image (Normalised)			-	-	-	-	1x28x28
Layer 1	Convolution Layer	Convolution + ReLU	32 -	3x3 -	1x1 -	1x1 -	28x28x32 28x28x32
Layer 2	Convolution Layer	Convolution + ReLU	32 -	3x3 -	1x1 -	1x1 -	28x28x32 28x28x32
Layer 3	Pooling Layer	Max Pooling	1	2x2	2x2	0	14x14x32
Layer 4	Convolution Layer	Convolution + ReLU	64 -	3x3 -	1x1 -	1x1 -	14x14x64 14x14x64
Layer 5	Convolution Layer	Convolution + ReLU	64 -	3x3 -	1x1 -	1x1 -	14x14x64 14x14x64
Layer 6	Pooling Layer	Max Pooling	1	2x2	2x2	0	7x7x64
Layer 7	Inner Product Layer	Fully Connected + ReLU	- -	- -	- -	- -	200

Layer 8	Inner Product Layer	Fully Connected + ReLU	- -	- -	- -	- -	200
Layer 9	Output Layer	Softmax	-	-	-	-	10

Table 1 – Base DNN Classifier

Note: each convolutional layer used a normal distribution for the initialisation of their respective filters.
The Keras library was used to train this classifier using the parameters listed in Table 2.

Parameter	Value
Learning Rate	0.1
Momentum	0.9
Dropout	0.5
Batch Size	128
Epochs	50
Decay	1e-6

Table 2 – Learning Parameters for Base DNN Classifier

The use of kernel or bias regularisation, or rather lack thereof, was not specified in the original reference materials. As a result, the choice to not apply kernel or bias regularisation to the base DNN model was taken. This choice increases the chance of large weights and biases being learnt by the model which increases the chance of a network to overfit to its data. These large weights and biases are typically ill-advised as they can then be exploited by adversarial attacks; however, due to the nature of the work being conducted, it offers a good method of comparison to how much a model can be improved via the defences proposed.

The fully trained model reported the metrics as stated in Table 3. Figure 2 and Figure 3 further illustrates this over each iteration. As a note, the metrics can seem odd at first interpretation due to the testing metrics being better than that of the training metrics. However, a model trained using the Keras library has two modes: training and testing. Any regularisation mechanisms such as the dropout used are not implemented at test time [16] hence can offer some reasoning for the training metrics being marginally worse than that of the testing metrics.

	Training Set	Testing Set
Cross Entropy Loss	0.0684	0.0462
Classification Accuracy	98.24	98.95

Table 3 – Summary of trained DNN Classifier metrics

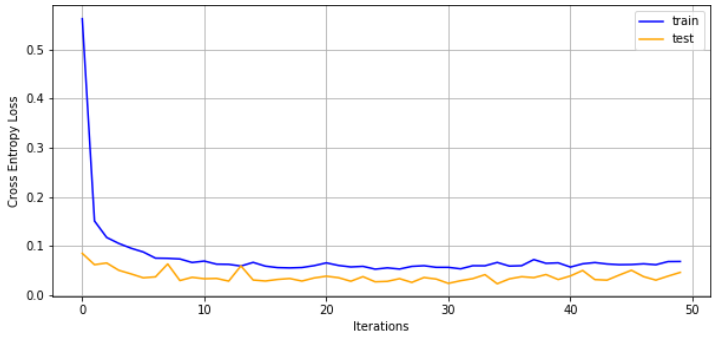

Figure 2 – Cross Entropy Loss

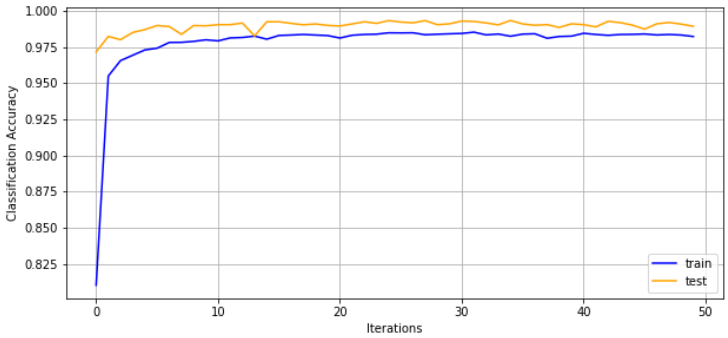

Figure 3 – Classification Accuracy

2.4 DEFENCE METHODS

Each defence architecture labelled below, makes use of multiclass Support Vector Machines trained on one or multiple features of the sublayers within the base DNN classifier and then consequently uses a rejection mechanism to discard possible perturbated samples. The specific layers each defence method is trained can be found in their respective sub-section.

Briefly, given training vectors $x_i \in \mathbb{R}^p, i = 1, \dots, n$ belonging to two classes and label vector $y \in \{1, -1\}^n$ (where n is the number of training data samples and p is the number of features in a data sample), the Support Vector Machine aims to construct a hyper-plane (or set of hyper planes in an infinite dimensional space) to complete the classification task which separates the two given classes accurately. A good separation of the samples from the hyperplane is achieved by maximising its distance $\left({}^{2}/_{\|w\|^2}\right)$ to the nearest training data point. This is then equivalent to the minimisation problem:

$$\min \quad \varphi(w) = \frac{1}{2}\|w\|^2 = \frac{1}{2}(w, w)$$
$$S.t: \quad y_i(w.x_i + b) \geq 1$$

Introduction of the Lagrange function then allows the conversion of the former minimisation problem into the following dual quadratic optimisation problem as similarly optimised by Scikit-Learn's SVC function:

$$\max \quad \sum_{i}^{n} \alpha_i - \frac{1}{2}\sum_{jk}^{n} \alpha_j \alpha_k y_j y_k (x_j \cdot x_k)$$
$$S.t: \quad \sum_{i=1}^{n} \alpha_i y_i = 0\,, \qquad 0 \leq \alpha_i \leq C$$

C is the regularisation constant which controls the trade-off between a smooth decision boundary and classifying the training points correctly. The regularisation constant, C was chosen through a process of grid search for a satisfactory cross validation score, such that C=0.1. After fitting, the hyperplane separating these two classes can then be defined by the decision function:

$$f(x) = sgn\left(\sum_{i}^{n} \alpha_i y_i (x_i \cdot x) + b\right)$$

Where α_i and b are the support vector components found for the optimum fit.

For non-linear cases, a kernel function $K(x_i, x)$ can be introduced to map the features into a higher dimensional feature space as to allow for a better fit of the hyperplane, where the decision function then becomes:

$$f(x) = sgn\left(\sum_{i}^{n} \alpha_i y_i K(x_i, x) + b\right)$$

The DNR and NR defence methods both make use of the RBF Kernel defined as such:

$$K(x, x_i) = \exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2}\right)$$

Where,

- $\|x\text{-}x_i\|^2$ is the squared Euclidean distance between the two feature vectors
- σ is a chosen constant

Scikit-Learn uses the following equivalent kernel which uses the parameter $\gamma = \frac{1}{2\sigma^2}$:

$$K(x, x_i) = \exp(-\gamma \|x - x_i\|^2)$$

All Support Vector Machines trained made use of $\gamma = 0.01$, where an increasing value of gamma leads to overfitting as the classifier attempts to perfectly fit the data. This was similarly chosen through a method of grid search as C.

A 'one-vs-rest' approach was taken in training the SVMs resulting in a separating hyperplane (and thus 10 separate SVMs) being trained for each of the 10 classes.

The "decision_function" method of Scikit-Learn's SVMs can give a per-class score for each sample by passing the sample (x) through the trained decision function. The output probabilities of each class ($s_0 \ldots s_9$) from the Support Vector Machines were then found using Scikit-Learn's "predict_proba" function which applies a logistic regression method on the SVM's decision function score.

2.4.1 SINGLE LAYER NEURAL REJECTION (NR)

The architecture proposed as by [15], involves the use of an RBF-SVM trained on the penultimate layer of the base DNN classifier as such illustrated by Figure 4. When training, the features from the relevant layer were extracted and used to train the SVM. The base DNN model remained unchanged during this training process. Note the probabilities of each class as the output, $s_0 \ldots s_9$, and the rejection mechanism governed by the individual rejection class (s_R) set to a constant value, θ: if s_R is the highest probability value, the sample is considered to have been rejected. The classification accuracy achieved by this SVM can be seen illustrated in Figure 7.

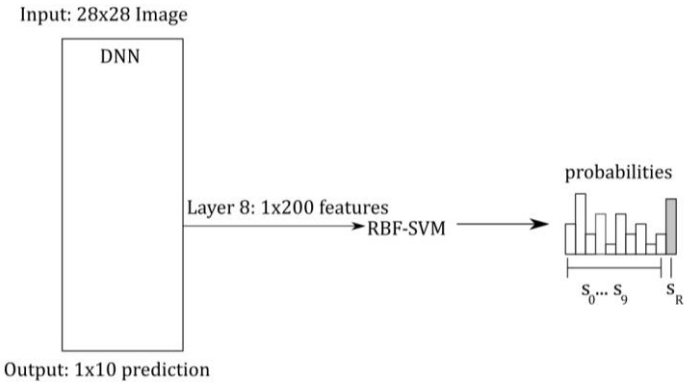

Figure 4 – NR Defence Architecture

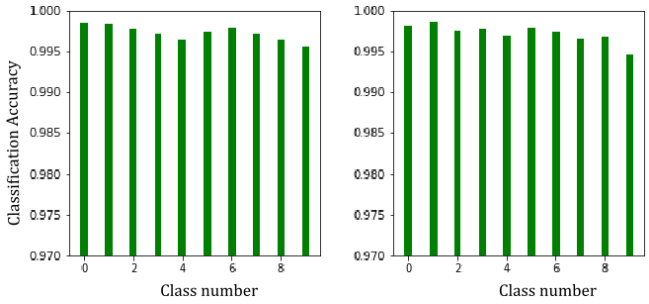

Figure 5 – Training (left) and Testing (right) Classification accuracy

2.4.2 DEEP NEURAL REJECTION (DNR)

Due to the effectiveness at reducing the impact of evasion attacks found by the defence proposed by [14], the combined use of multiple layer's features to better recognise perturbations was proposed by [13] as illustrated by Figure 6. The training of each SVM was completed similarly to that of NR however the features from layer 5 needed to first be flattened into a vector prior to being input into its respective SVM. Predictions from these SVM models were then concatenated into a 1x30 vector which was consequently fed into the final SVM model for training. Similar to previous, each model was trained individually of each other. Once again, note the rejection mechanism implemented in the same manner as in the NR architecture, but only in the final SVM.

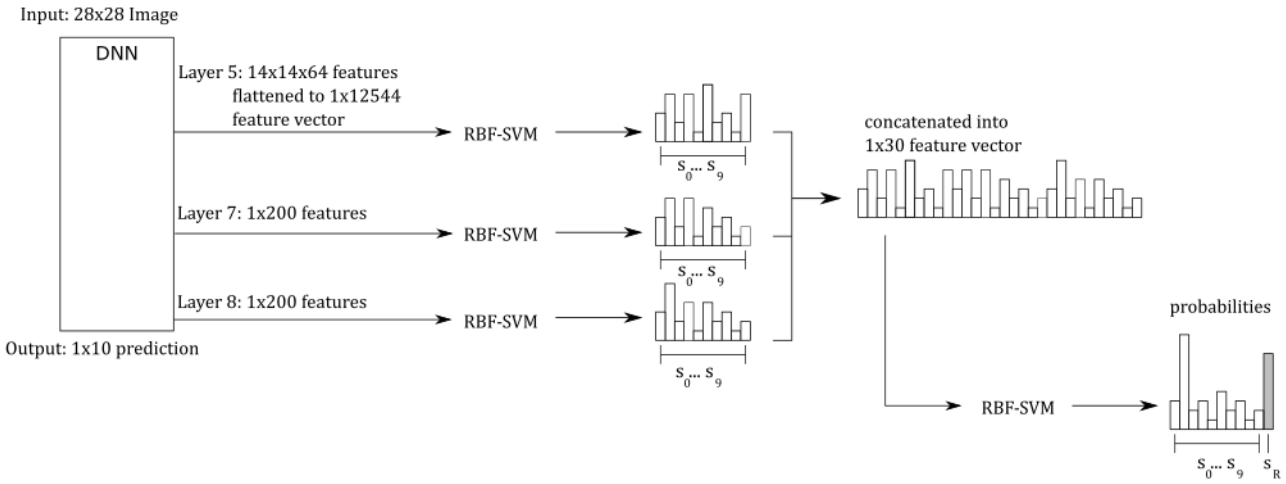

Figure 6 – DNR Defence Architecture

The accuracies achieved by each of the 3 SVMs ('sub-SVMs' trained directly on the extracted features can be seen illustrated in Figure 7. The accuracy achieved by the final concatenated SVM can be seen illustrated in Figure 8.

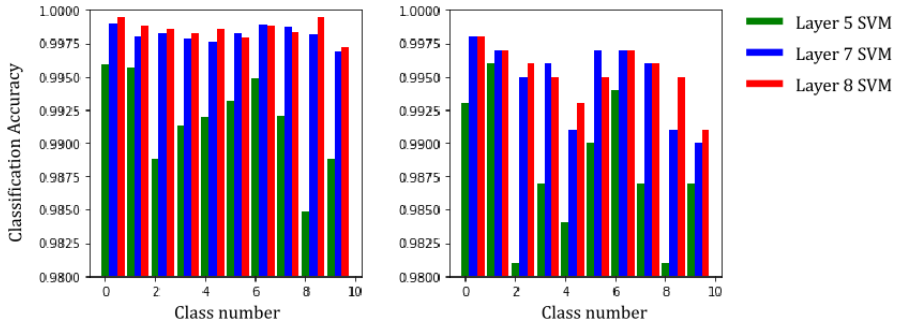

Figure 7 – Training (left) and Testing (right) Classification accuracy for each individual 'sub-SVMs'

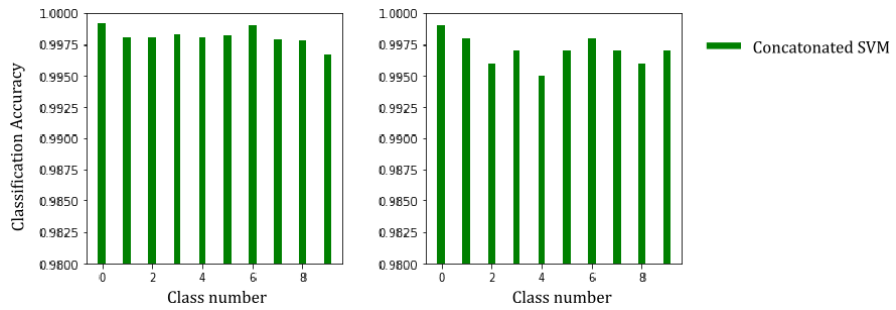

Figure 8 – Training (left) and Testing (right) Classification accuracy of final concatenated SVM

2.4.3 ADVERSARIAL LEARNING

As seen implemented by such defences as [7][8], the use of adversarial learning is the other commonly implemented strategy in the mitigation of successful adversarial attacks. This is on the basis that adversarial images exist due to a lack of regularisation of the model as suggested by [17] and exploit 'pockets' of untrained space within a classifier. Adversarial learning aims to retrain the classifier using generated adversarial examples thus 'filling' these pockets of space lacking training, consequently improving the generalisation of the model.

This was completed as a two-step process where the original classifier was similarly attacked to the models above, and the successful adversarial examples used to retrain the original model. This original model was then re-attacked to observe the effectiveness of the consequent attack as illustrated by Figure 9.

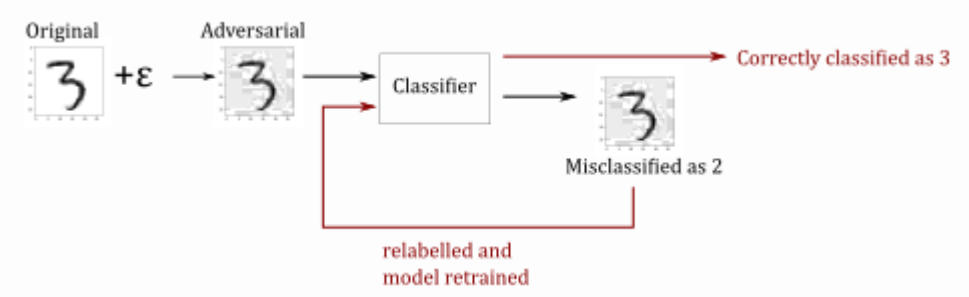

Figure 9 – Adversarial Training Process

2.5 ATTACK METHODS

An attack on a deep neural network is directly dependant on the level of the attacker's knowledge of said model. They can essentially be separated into two general categories:

- A PK attack – a scenario where the adversary has perfect knowledge of the targeted classifier. As a result, the attacker can be assumed to have access to all model parameters such as its gradients, but is unable to physically change the model and does not have access to the training data.
- A LK attack – a scenario where the attacker has a limited knowledge of the targeted classifier.

Intuitively, the more knowledge that an adversary has of the model, the greater effectiveness of the attack as has been demonstrated by multiple experiments. [18][19]. For the purpose of thoroughly testing defence techniques, a 'defence-aware' gradient attack will be implemented where the attacker will have a perfect knowledge (PK) of our defence architecture.

2.5.1 EVASION ATTACK METHOD

The adversary's goal is to take any sample x_0 and through an attack strategy find a sample x_{adv}, where:

$$x_{adv} = x_0 + \varepsilon$$

$$S.t: \varepsilon \geq \|x_0 - x_{adv}\|_2$$

The process of constructing an adversarial example can be thought of as solving the following constrained optimisation problem:

$$\min \quad (\max(s_i) - \max(s_j))$$
$$S.t: \quad i \in \{y, r\}, \qquad j \notin \{y, r\}$$

Where s represents the probability of a class.

Given that a decision on the classification of a sample is made by whichever classification probability is the largest, it can be understood that to create an adversarial example, one must minimise the probability of the correct class, s_i, and consequently increase the probability of the next highest incorrect class, s_j. The rejection constant (s_r) was used in place of the probability of the correct class, in the case that $s_r > s_i$. By then minimising the distance $\max(s_i) - \max(s_j)$, the most effective adversarial example can be constructed. For the tests conducted, $s_r = 0.9$.

The value, $\max(s_j)$, indicates the maximum of any of the other class probabilities, which leads the nature of the attack to being inherently untargeted. This attack can be adapted to perform a targeted attack by replacing this variable with the probability of another specific class that one would wish to cause the classifier to misclassify to. However, for the purpose of solely testing the robustness of a defence, this was decided unnecessary.

Through an iterative process, this optimisation to minimise the score of the true class and maximise a false class can be solved by making small perturbations to the input image. A successful attack will therefore be one which is able to successfully manipulate an input image, whilst being within the L2 boundary (ε), to cause a misclassification.

2.5.2 PROJECTED GRADIENT DESCENT (PGD)

It was possible to implement this evasion attack through a process of Projected Gradient Descent (PGD). Given an example image (x_0) with label y_0 and probability s_0 as such:

$$for\ x_0 \Rightarrow \begin{matrix} label\ y_0 = [-1,-1,-1,+1,-1,-1,-1,-1,-1,-1] \\ probability\ s_0 = [0.1, 0.5, 0.01, 0.9, 0.05, 0.12, 0.16, 0.21, 0.05] \end{matrix} \Rightarrow 3$$

Its adversarial target can be obtained (the second highest probability prediction) and consequently an adversarial label made to match this target. It can be seen that for x_0 with $y_0 \Rightarrow 3$, the second highest prediction is for class 1. Hence an adversarial label, y_{adv}, can be made as such for the intended adversarial example x_{adv}:

$$for\ x_{adv} \Rightarrow adversarial\ label, y_{adv} = [-1,+1,-1,-1,-1,-1,-1,-1,-1,-1]$$

Hinge loss is typically used for multiclass classification applications, and is notably used in support vector machines. It can be detailed as such:

$$Loss, L = \max(0, 1 - y_i(f(x_i))$$

$$\because f(x) = sgn\left(\sum_i^n \alpha_i y_i K(x_i, x) + b\right)$$

For the purpose of creating an adversarial example, this loss function instead made use of y_{adv} in place of y_i for x_{adv} with the objective of reducing the loss for the adversarial label. This results in the following modified loss function:

$$L = \max(0, 1 - y_{adv}(f(x_{adv}))$$

Through gradient descent, this loss could then be minimised by applying small perturbations to the input image through an iterative process as such:

$$x_{k+1} = x_k - \eta \nabla L$$

If x_{k+1} exceeds the l2-norm distance (ε) that is predefined by the utilised constraint, it is projected back inside as so:

$$x_{k+1} := x_0 + \varepsilon(x_{k+1} - x_0)/\max(\varepsilon, \|x_{k+1} - x_0\|_2)$$

This process of creating an adversarial example can be more succinctly described by Algorithm 1 below.

Algorithm 1: PGD Based Attack Procedure

Input The initial sample, x_0 ; the step size, α; the l2 norm distance, $\epsilon = \|\mathbf{x} - \mathbf{x_{adv}}\|_2; t > 0;$

Output The adversarial example, x_{adv}

$x_{adv} \leftarrow x$

Initialise noise and add to input sample (x) whilst remaining inside L2 norm : x_k;

repeat

- $x_{k+1} \leftarrow x_k - \eta \nabla L(x_k)$;
- $p \leftarrow x_{k+1} - x_0$;
- $x^{k+1} \leftarrow x_0 + \frac{\epsilon(p)}{max(\epsilon, \|p\|_2)}$;

until $L(x_{k+1}) - L(x_k) \leq t$;

Note that the constant 't' was an arbitrarily chosen constant as to confirm a level of convergence to the algorithm. In the tests conducted, it was found sufficient for t=0.01.

The gradient of the loss function w.r.t to the input image was chosen to give an indicator of the direction of perturbation with the aim of reducing this loss. The gradient of the Neural network portion of the architecture in question could be calculated simply using the Gradient Tape method provided by TensorFlow's API: a method of automatic differentiation. Using techniques such as the chain rule, it was then possible to combine this with the gradient of the SVM portions of the architecture to get the overall gradient of the output loss w.r.t input image so that a perturbation could be applied. This gradient for the SVM portions could be calculated as such:

$$\because Loss, L = \max(0, 1 - y_i(f(x_i))$$

$$\Longrightarrow \frac{dL}{dx_i} = \begin{cases} 0 & if\ y_i f(x) > 1 \\ -y_i f'(x) f(x) & if\ y_i f(x) < 1 \end{cases}$$

For the non-zero case, the gradient directly correlates to the gradient of the decision function.

$$f(x) = \sum_i^n \alpha_i y_i K(x_i, x) + b$$

$$\Rightarrow f'(x) = \sum_{i}^{n} \alpha_i y_i \nabla K(x_i, x)$$

We see that this gradient is then directly related to the gradient of the Kernel. For an RBF Kernel, then:

$$K(x_i, x) = \exp(-\gamma \|x - x_i\|^2)$$
$$\Rightarrow \nabla K(x_i, x) = -2\gamma \exp(-\gamma \|x - x_i\|^2)(x - x_i)$$

3. RESULTS

As mentioned in Section 2.1, the adversarial attack was used to create a dataset of adversarial examples. Examples of the adversarial images created for each defence method can be seen illustrated in Figure 10. This was then used as to extract the performance of each defence method. When considering the accuracy of the rejection based methods mentioned, a rejection of a sample under perturbation was also considered a successful classification. For both the rate of rejection and accuracy of each architecture, see Figure 11.

The DNR method can be seen to perform to a similar degree until ε=3, at which point we can see the DNR defence method begins to out-perform. The rejection rate can be seen to follow a similar correlation, albeit not as severe. When performing a comparison between the research conducted by [13], which this report tries to follow as closely as possible, and the results found here, a similar correlation between the three architectures can be seen.

However, the attack conducted here, shows a far weaker response which can be a result of the attack itself not being as well optimized as possible. This apparent difference in attack strengths can also be reasoned to a difference in classifier and attack parameters that may have been used. The use of a larger grid search, similar to that used when training the SVMs, may offer some insight into more optimum parameters that could be instead used.

Some research has also used extra components in addition to the loss function such as a density estimator. The use of such estimators penalise perturbations in low density regions and have been shown to favour attack points which imitate features of known samples such as those in mimicry based attacks [20][21]. The adversarial samples produced in [15] show evidence of much higher resolution noise in addition to the features of other classes being replicated which add to the possibility of such an estimator having been used. In contrast, the nature of the adversarial images generated here seem to lack this element of density. The difference in nature of the adversarial examples generated between the two reports offer opportunity to improve the work done here in the future as to perform an even more severe attack through some of the methods discussed.

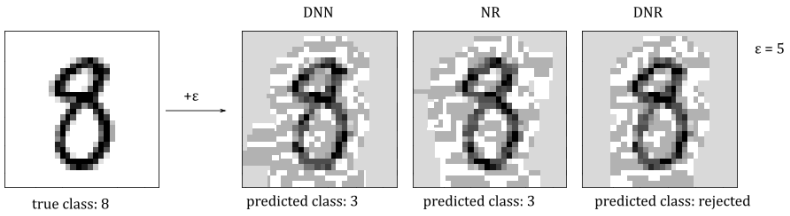

Figure 10 – Adversarial examples generated

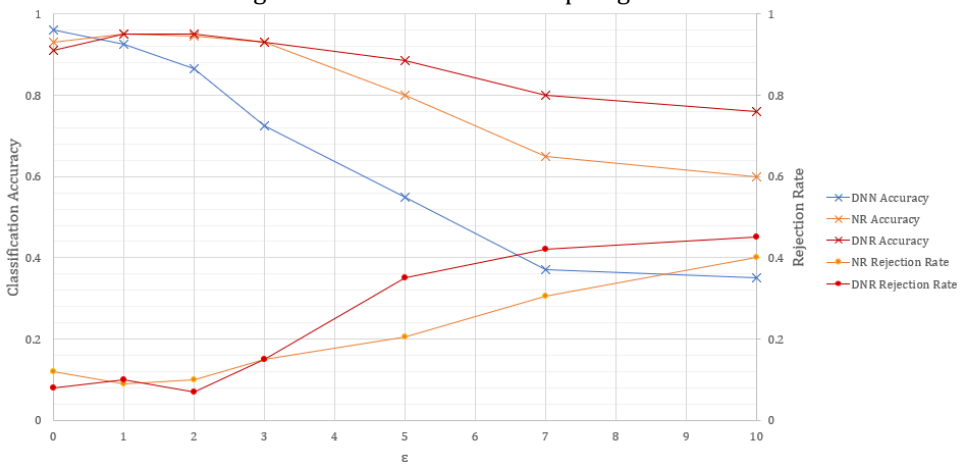

Figure 11 – Classification Accuracy and Rejection Rate results

Due to the good performance demonstrated by this DNR architecture, the techniques employed in adversarial training were consequently implemented on the defence. This was implemented as a means of understanding the effectiveness of combining two popular schools of thought involving the defence against adversarial examples. The base DNN was also retrained as to offer a better comparison between the methods.

When performing adversarial training, only the adversarial images which were able to cause a successful misclassification were used. Consequently, a newly created adversarial dataset was used to report the accuracy and rejection rate metrics seen in Figure 12. Despite the additional training applied, the results were, for the most part, unchanged. Both the DNR and undefended DNN models showed marginal improvement throughout, particularly at the larger values of ε. However, it is unknown whether this is instead a result of random initialisations that exist throughout the process of training models and generating adversarial images: some variation is naturally expected between different training cycles. The small sample size of adversarial images added to the original datasets for retraining would also undoubtedly be to some extent responsible for the lack of impact on the results seen. Given the computational expensiveness of the process of generating adversarial images and consequently retraining the model this way, it was impractical to produce an additional dataset the size of which would be able to impact the current results.

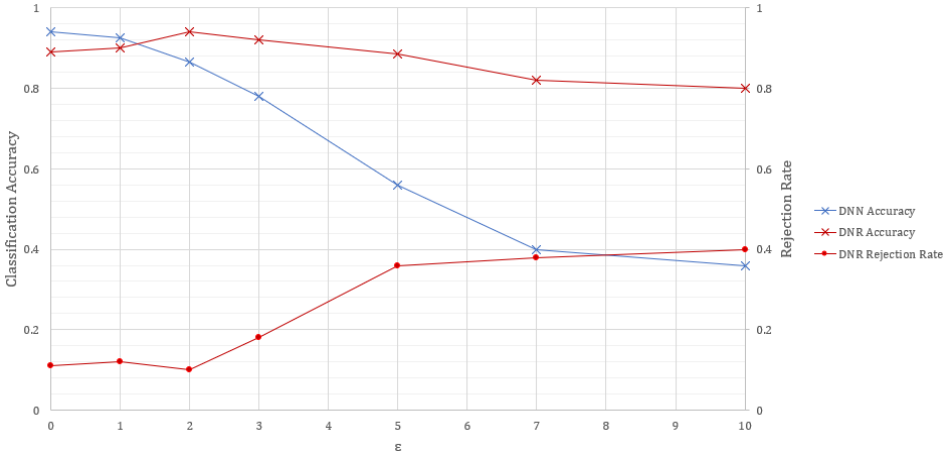

Figure 12 – Classification Accuracy and Rejection Rate after Adversarial Training

4. CONCLUSION AND FUTURE WORK

In this work we have investigated the effectiveness of existing methods of making a classification more robust. We have in particular been attracted to the simplicity of the Deep Neural Rejection method of adversarial defence. The benefits of using earlier computed features in an existing network as to better detect perturbations show great improvement over the previously proposed Neural Rejection method, which only utilises the features of the penultimate layer to form a decision.

We have tested the effectiveness of adversarial training; nevertheless, due to the computational expensiveness of the method proposed, creating large enough of a dataset to have an impact on the learnt features was not practical. Given enough additional datapoints made of adversarial data, one would expect the accuracy of the trained model to be improved at larger values of ε. However, one would also expect a lower accuracy for unperturbed images. Despite this expectation, it was not possible to complete this at this given opportunity. In comparison to this attempt, DNR by itself, proved far less computationally expensive of a defence method to employ and showed a very promising accuracy and ability to withstand adversarial examples even at high levels of perturbation.

In addition, given the disparity in the resolution of noise between the perturbations created in this work and that which is observed in [13], it would also be interesting to investigate whether the use of a kernel density estimator would allow for more precise perturbations and consequently a more severe adversarial attack in future work. Another direction that offers room for exploration is the computational expensiveness of the adversarial training process; it would be interesting to tackle this via the use of a Generative Adversarial Network (GAN) as investigated in [22] which offers the possibility to reduce the expensiveness of the operation and allow for the generation of a much greater dataset of potential adversaries. This would then allow for a better comparison between the two general approaches to defending against adversarial examples and may provide some further insight.

5. REFERENCES

[1] X. Wang, T. Qiu, C. Chen and N. Chen, "A Neural-Network-Based Real-end Collision Prediction Mechanism for Smart Cities," in *2019 IEEE International Conference on Smart Internet of Things (SmartIoT)*, Tianjin, China, 2019, pp. 496-497.

[2] C. Sowmiya and P. Sumitra, "Analytical study of heart disease diagnosis using classification techniques," in *2017 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS)*, Srivilliputhur, 2017, pp. 1-5.

[3] A. S. Katasev and D. V. Kataseva, "Neural network diagnosis of anomalous network activity in telecommunication systems," in *2016 Dynamics of Systems, Mechanisms and Machines (Dynamics)*, Omsk, 2016, pp. 1-4.

[4] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, "Intriguing properties of neural networks," 2013. arXiv:1312.6199.

[5] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, D. Song, "Robust Physical-World Attacks on Deep Learning Models," 2018. arXiv:1707.08945

[6] N. Carlini and D. Wagner, "Audio Adversarial Examples: Targeted Attacks on Speech-to-Text," in *2018 IEEE Security and Privacy Workshops (SPW)*, San Francisco, CA, 2018, pp. 1-7.

[7] J. Wang and H. Zhang, "Bilateral Adversarial Training: Towards Fast Training of More Robust Models Against Adversarial Attacks," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South), 2019, pp. 6628-6637.

[8] A. Bagheri, O. Simeone and B. Rajendran, "Adversarial Training for Probabilistic Spiking Neural Networks," in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Kalamata, 2018, pp. 1-5.

[9] I. Goodfellow, J. Shlens, C. Szegedy, "Explaining and Harnessing Adversarial Examples", 2014. arXiv:1412.6572

[10] M. Esmaeilpour, P. Cardinal and A. Lameiras Koerich, "A Robust Approach for Securing Audio Classification Against Adversarial Attacks," in *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2147-2159, 2020.

[11] M. Naseer, S.H. Khan, F. Porikli, "Local Gradients Smoothing: Defense against localized adversarial attacks,"2018. arXiv:1807.01216

[12] A. S. Ross, F. Doshi-Velez, "Improving the Adversarial Robustness and Interpretability of Deep Neural Networks by Regularizing their Input Gradients," 2017. arXiv:1711.09404

[13] A. Sotgiu, A. Demontis, M. Melis, B. Biggio, G. Fumera, X. Feng, F. Roli, "Deep Neural Rejection against Adversarial Examples," 2020. arXiv: 1910.00470

[14] M. Melis, A. Demontis, B. Biggio, G. Brown, G. Fumera, and F. Roli, "Is deep learning safe for robot vision? Adversarial examples against the iCub humanoid," in *ICCVW Vision in Practice on Autonomous Robots (ViPAR)*, pp. 751–759, IEEE, 2017.

[15] N. Carlini and D. A. Wagner, "Towards Evaluating the Robustness of Neural Networks," 2017. arXiv: 1608.04644

[16] Keras FAQ: Frequently Asked Keras Questions [Online]. Available: https://keras.io/getting-started/faq/#why-is-the-training-loss-much-higher-than-the-testing-loss [Accessed: 17 April 2020]

[17] P. Tabacof, E. Valle, "Exploring the Space of Adversarial Images," 2015. arXiv:1510.05328

[18] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, P. Abbeel, "Adversarial Attacks on Neural Network Policies," 2017. arXiv:1702.02284

[19] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, A. Swami, "Practical Black-Box Attacks against Machine Learning," 2016. arXiv:1602.02697

[20] B. Biggio, I. Corona, B. Nelson, B. Rubinstein, D. Maiorca, G. Fumera, G. Giacinto, F. Roli, "Security Evaluation of Support Vector Machines in Adversarial Environments," 2014. arXiv:1401.7727

[21] P. Fogla, M. Sharif, R. Perdisci, O. Kolesnikov, W. Lee, "Polymorphic blending attacks," in *Proceedings of the 15th Conference on USENIX Security Symposium,* 2006.

[22] J. Thanthulage, 'Robust Image Classification using Deep Learning – Research Report,' Dept. Elec. Elect. Eng., Loughborough Univ., Project Rep., 2019.