Calibration of neural network logit vectors to combat adversarial attacks.

Oliver J Goldstein*[1]

[1]Department of Computer Science, Bristol University

Master's Thesis: 2017 - 2018

*oliverjgoldstein@gmail.com, http://www.oliverjgoldstein.com

Contents

1 A brief introduction to calibration **8**

1.1 Examples of calibration methods . . . 10

1.1.1 Binning . . . 10

1.1.2 Isotonic regression . . . 10

1.1.3 Platt scaling . . . 11

1.1.4 Logistic regression . . . 11

1.1.5 Softmax . . . 13

1.1.6 Temperature scaling . . . 14

1.2 Beta calibration . . . 15

1.3 Beta calibrations relationship to temperature scaling . . . 16

1.4 Conclusions . . . 19

2 Adversarial attacks **20**

2.1 Types of adversarial attack . . . 20

2.2 Adversarial attack methods . . . 21

2.2.1 L-BFGS . . . 21

2.2.2 FGSM . . . 21

2.2.3 Basic Iterative Attack . . . 22

2.2.4 JSMA . . . 22

2.2.5 DeepFool . . . 23

2.3 Conclusions . . . 23

3 Defences against adversarial attacks **24**

3.1 Proactive defences . . . 24

3.1.1 Network distillation . . . 24

3.1.2 Why does defensive distillation work? . . . 24

3.2 Reactive defences . . . 24

3.2.1 Neural network training as a defence . . . 24

3.2.2 Statistical defences . . . 25

3.3 Conclusions . . . 26

4 Background Check **27**

4.1 How does Background Check work? . . . 28

4.1.1 Justification of $q_b(x)$ . . . 28

4.1.2 Performance of Background Check . . . 29

4.2 Conclusions . . . 29

5 Methodology **30**

5.1 How is Background Check used? . . . 30

5.2 Architecture . . . 33

6 Results **34**
6.1 Visualisation of results . . . 34
6.2 Description of images . . . 34
6.3 Table of results . . . 39
6.4 Pipeline of code and data analysis . . . 40
6.4.1 Materials and Methods . . . 40
6.4.2 Why this particular architecture? . . . 41
6.4.3 Parameter choices: Large, Typical, Small, Very Small . . . 41
6.5 Separating images based on the relative difference of logits . . . 42
6.6 Discussion of results . . . 42
6.7 Evaluation of Background Check in the context of adversarial defences . . . 45
6.7.1 Disadvantage of methodology and caveat of results . . . 45
6.7.2 Advantage of methodology . . . 49

7 Analysis: Limitations of neural networks **50**
7.1 Does high accuracy mean that the logits are Gaussian? . . . 51
7.2 Commentary: Why are accurate probabilities fundamentally hard to guarantee? . 52
7.3 Generative versus discriminative approaches . . . 53
7.3.1 Why is it important to know what you don't know? . . . 55
7.4 Impact - i-LIDS - An example of adversarial risk in the public sector . . . 55

8 Future work **57**

9 Conclusion **57**

Appendices **59**

A Names of attacks **59**

B Parameters **59**

C Supplementary Figures **59**

Author's Declaration

I declare that the work in this thesis was carried out in accordance with the requirements of the University's Regulations and Code of Practice and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the thesis are those of the author.

SIGNED: OLIVER GOLDSTEIN..... DATE: 4th Sept 2018.....

List of abbreviations

1. **FGSM** - Fast Gradient Sign Method
2. **JSMA** - Jacobian Saliency Map Attack
3. **L-BFGS** - Limited-memory Broyden Fletcher Goldfarb Shanno
4. **BI** - Basic Iterative
5. **VAT** - Virtual Adversarial Training
6. **Mom** - Momentum
7. **BCD** - Discriminative approach
8. **BCF** - Familiarity approach
9. **CWA** - Closed world assumption
10. **i-LIDS** - The Image Library For Intelligent Detection Systems
11. **logit vector** or **logits** - output vector before the softmax layer.
12. **DVP** - Dog versus Plane (NB as in airplane)
13. **AVH** - Airplane vs Horse
14. **FVS** - Fish vs Ship
15. **CIFAR** - Canadian Institute For Advanced Research
16. **TPR** - True Positive Rate

Abstract

The accuracy of many classification tasks, such as image recognition, has improved massively in the last decade. This improvement has mostly been caused by the scaling and improvement of artificial neural networks (Huang et al. 2017). Model calibration and refinement, two measures that have had less attention paid to them, determine the quality of frequentist probability estimates of models[1]. In contemporary papers (Sabour, Frosst, and G. E. Hinton 2017), (Lyksborg et al. 2015), (Gkioxari, Girshick, and Malik 2015), (Zeng 1999) the application of the softmax function is said to produce probabilities from the logit vectors of neural networks. This thesis provides some theoretical and empirical understanding of these probability estimates in the context of neural networks. An example of context is the ease with which supposedly robust probability estimates are subverted by adversarial attacks, a recently discovered phenomenon (Szegedy et al. 2013) where examples can be subtly manipulated such that they are mis-classified by neural networks. Adversarial attacks are therefore used as a case study in the context of this thesis.

The central question this thesis provides insight into, is whether Background Check (Perello-Nieto et al. 2016), a novel calibration technique, can assist neural networks in classifying adversarial examples, using the one dimensional difference between logit values as the underlying measure. To answer this question, many two-class neural networks, with a final softmax layer, are created, that achieve high average recall ($\sim$91%), equation (36), on test data. Adversarial attacks from various classes are then generated and correspondingly defended against with defence based on Background Check. This method successfully defends against various adversarial attack types, especially attacks that are generated with large attack parameters, which to date haven't been successfully defended against in contemporary literature. The proposed method does not need knowledge of the attack parameters or methods at training time, unlike a great deal of the literature, that imbues the proposed method with a great deal of flexibility. The new proposed method in this thesis can also detect individual adversarial examples. After doing a literature review, I realised that this is a completely new way of defending against adversarial attacks. Additional material provided includes an analysis of the difference between generative and discriminative classifiers, a demonstration that the logits are not necessarily Gaussian, even with high average recall, and a proof relating the expressive power of two different calibration methods, temperature scaling and beta calibration.

I initially investigated calibration in the context of neural networks, before questioning why simple calibration methods (for instance, calibration that relies on modifying the gradient of the softmax function, such as (Guo et al. 2017)) were being researched in contemporary literature. The reason I questioned why these methods were used was because far more

[1] (Cosmides and Tooby 1996) & (Vallverdú 2015) discuss frequentist and bayesian notions of probability theory, respectively

complex issues exist, in the context of neural network performance, specifically the spectre of adversarial attacks. Given a basic understanding of adversarial attacks, I hypothesised that the adversarial examples would appear in regions in which no training data had been seen and so looked for ways to model the adversarial regions. In addition, much of the analysis in chapter 7 and all of lemma 1.1 - lemma 1.3 is original work.

Structure of this thesis

The history of calibration, adversarial attacks and adversarial defences are explored in chapters 1, 2 and 3 to give context to later analysis. In chapter 1, a proof demonstrates the difference between 2-parameter beta calibration and temperature scaling. In chapter 4, the Background Check method, which underlies the classification of adversarial examples is covered to give context to chapter 5. In chapter 5, the methodology used to create neural networks is discussed. In chapter 6, the tabulated results, histograms and are followed up by data analysis, a justification of the architecture type and an evaluation of the method taken, to defend against adversarial attacks. In chapter 7, a deeper analysis is given on why adversarial attacks happen along with an empirical result showing that high average recall does not necessarily entail normally distributed logit vectors. A discussion on generative and discriminative classifiers is also given. Chapter 8 provides future work and finally, chapter 9 concludes on the key points of the thesis.
The main contributions this thesis brings are listed here.

1. An identification of novel links between the fields of calibration and adversarial defences.
2. A proof demonstrating the difference between temperature scaling and 2-parameter beta calibration.
3. A novel method to defend against many adversarial attack types. It is very different to previous attacks, in that, images perturbed with a *large attack vector* are more easily found than those with small perturbation vectors. The method also has an accompanying critical evaluation.
4. An empirical evaluation of the normality of logit distributions on a high average recall neural network.
5. A proposed extension to the temperature scaling calibration procedure.

1 A brief introduction to calibration

Daniel Kahneman intuitively explains calibration:

> *"if you give all events that happen a probability of .6 and all the events that don't happen a probability of .4, your discrimination is perfect but your calibration is miserable"*

A classifier is said to be well calibrated, if, as the number of predictions approaches infinity, the proportion of outcomes given probability p, occur p fraction of the time.

Denoting $x_1, x_2, ..., x_n$ as dataset instances and $y_1, y_2, ..., y_n$ as their corresponding ground truth class labels, a scoring classifier $s = f(x_i)$ has a calibrating function μ applied to it yielding $\mu(f(x_i))$. Perfect calibration is defined as the expectation $s_i = f(x_i)$ such that $s_i = \mathbb{E}[Y|f(X) = s_i]$ where random variables X, Y denote the features and class label of a uniformly randomly drawn instance from the dataset respectively, such that $Y = 1$, $Y = 0$ represent an individual positive and negative instance, respectively.

Calibration of machine learning models, is normally achieved by a post-processing map "on top of the model", i.e. model scores are converted into probabilities through a calibrating function. Model and data assumptions are used to construct a mapping from scores to frequentist probabilities. These score dependent class ratios are interpreted as posterior frequentist probabilities, i.e. that an instance is a member of particular class i, $P(y_i|x)$.

An informal example of perfect calibration, can be described in a possible world where an event is either of class x or y with uniform probability of one half. This can be formally written as $\forall s, C = (x_i, y_i)\ \mu(s) = 1/2$, where s is the score of an instance from C_i.

A natural objection to the approach of placing a calibration map on top of the model, is, rather than post-processing model scores with a calibration map, to assert, that it is better to fit the optimization method with an objective function that coerces models to produce decision boundaries, such that the scores are already a reflection of well calibrated probability estimates ab initio. One paper that adopted this approach was (Menon et al. 2012) which combined Isotonic Regression, a calibration method that works well when model scores are well ranked, with an optimization method based on increasing the AUC, the probability that a randomly drawn positive example has a score higher than a negative example. However, this method, only achieved minor improvements on probability estimates against standard logistic regression (Cox 1958). This may be due to the difficulty of finding such a decision boundary where the distance is a good predictor of the likelihood ratio. However, no more work in this area has been done. A common approach in calibration, is to increase the expressive power of the calibration map if the score distribution type is unclear, allowing the map to fit the underlying distribution.

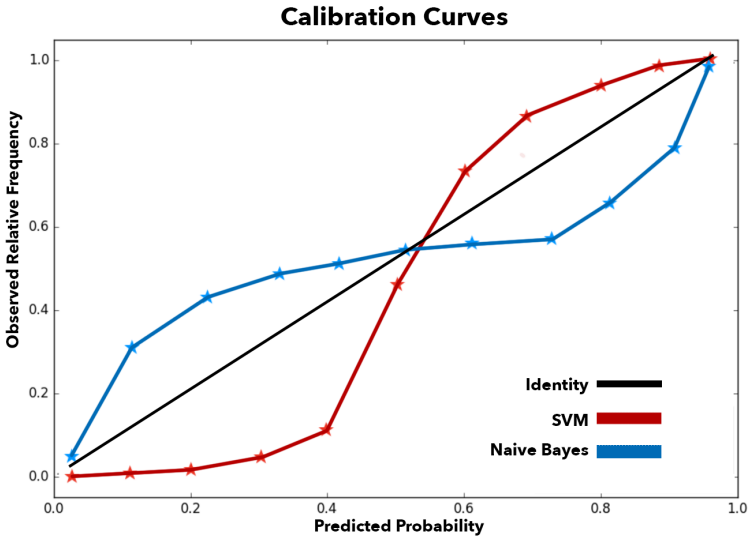


Figure 1: Calibration curves for a variety of different models on Gaussianly distributed data with two classes. Optimal calibration performance occurs when the predicted probability is equal to the observed relative frequency for all probability values, i.e. the curve exactly fits the identity line. A point with an x co-ordinate lower than the corresponding y co-ordinate, means a classifier is over-confident for predictions associated with that probability and vice versa. Naive Bayes, outputs probabilities directly, assuming feature independence in the process. As this assumption is rarely met, given the classifier predicts a low probability for a particular class, it can be more likely to be the other class. This occurs also for probabilites closer to one, in the opposite fashion. The typical SVM distortions can be explained as follows: One can imagine two classes, each with a Gaussian distribution. As the mapping from scores to probabilities is typically done linearly, according to (1), examples of data, at locations corresponding to 20 percent of the maximum probability value, well within the first cluster, the classifier will predict a predicted probability of 0.2, yet the data will most likely always be the first class. Once again, this also occurs for probabilites closer to one, in the opposite way.

To visualise calibration performance of a classifier, (Murphy and Winkler 1977) plots the observed frequency of an event against the predicted frequency yielding calibration curves. An example of calibration curves can be seen in in Figure 1.

This can be combined with a frequency distribution, giving an indication of spread, which is useful if there are few events associated with a particular probability. The notion of refinement is also useful when considering calibration. By considering the crude constant classifier, which predicts the probability corresponding to the class distribution for all model outputs, it is clear that this calibration estimate is perfectly calibrated. However, an intuitively more valuable calibration estimate, is one which predicts a value closer to either 0 or 1. For this reason, (DeGroot and Fienberg 1981) suggests measuring refinement loss, which measures the distance of the classifiers probability estimates to either 0 or 1[2]. Together, calibration and refinement

[2]Presumptuous probability ratings are associated with far larger error metrics if used with log loss

loss make up the Brier Score. (Kull and Flach 2015) defines calibration and refinement loss.

Calibration loss = $\mathbb{E}[d(S, C)]$ is the loss due to the expected difference between the model score S and the proportion of positives among instances (observed relative frequency) with the same score.

Refinement loss = $\mathbb{E}[d(C, Y)]$ is the loss due to the presence of instances from multiple classes among instances with the same estimate S. In the worst case, this clearly reduces to the crude constant classifier mentioned above.

1.1 Examples of calibration methods

Calibration has been approached from different angles, of which four of the most well-known are detailed here: Binning, Isotonic Regression, Logistic Regression & Platt Scaling.

1.1.1 Binning

Binning (Zadrozny and Elkan 2001) is a non-parametric calibration method, i.e. it does not assume anything about the underlying distribution of scores. Consequently, it is less model specific than other calibration procedures mentioned. However, unsurprisingly, the flexibility that it affords also costs this method.

During training, instances are placed into bins, depending on their score. In order to retrieve the probability of a test instance, the score relevant bin of the test instance is found and its relative class distribution in that bin wrt. training instances, is given as the probability estimate for each class. If there are too few bins, (one can imagine a single bin) then the confidence estimate is weak and decomposes to the class ratios, which has high refinement loss. Alternatively, if there are too many bins then there will be no data in each bin at training time, yielding probability estimates that will tend to 0. For this reason, the number of bins used is a hard parameter to set. In the literature cross validation sets are used to set this parameter, but even this involves issues such as unbalanced data sets or whether or not the bins cross over volatile regions.

1.1.2 Isotonic regression

Isotonic regression is a non-parametric method, (Zadrozny and Elkan 2002) that combines sigmoid fitting and binning. It is implemented with the pair-adjacent-violators (PAV) algorithm (Ayer et al. 1955). The key assumption that underlies this method, is that the score function is non decreasing and the mapping from scores to probabilities is non decreasing, which is a characteristic of sigmoid functions. The PAV algorithm chooses bin boundaries and sizes based on how well the classifier ranks examples. (Niculescu-Mizil and Caruana 2005) notices that this method works better than Platt Scaling after around 1000 data points in the calibration set.

1.1.3 Platt scaling

Previously to Platt Scaling, the distances of the point to the decision hyperplane was interpreted as a score in a linear manner. In particular, if the range of scores is $[b, -b]$ and $f(x_j)$ is a particular score, then obtaining a corresponding probability is done by adding the absolute minimum value, as shown in (1) and then normalising to $[0, 1]$ through division by $2b$. Platt (Platt et al. 1999) applies logistic regression to scores from an SVM model. The parameters from (2), β_0, β are found via maximum likelihood estimation on training sets of per-class probabilities.

$$s(x_j) = \frac{f(x_j) + b}{2b} \tag{1}$$

$$P(C_j|x_j) = \frac{1}{1 + e^{\beta_0 + \beta x_j}} \tag{2}$$

1.1.4 Logistic regression

The importance of understanding logistic regression from first principles is important when considering the limitations of softmax later on.

There are at least two ways to deduce logistic regression[3], a method originally discovered by (Cox 1958). One perspective is shown by (3), where there are two classes C_0, C_1 and data points denoted by x_j.

$$\begin{aligned} P(C_0|x_j) &= \frac{P(x_j|C_0)P(C_0)}{P(x_j|C_0)P(C_0) + P(x_j|C_1)P(C_1)} \\ &= \frac{1}{1 + \exp(-\log(\frac{P(x_j|C_0)}{P(x_j|C_1)}) - \log(\frac{P(C_0)}{P(C_1)}))} \end{aligned} \tag{3}$$

where (3) is derived through the application of the law of total probability, followed by dividing by $P(x_j|C_0)P(C_0)$, and then using the fact that exponential terms cancel logs, then trivially flipping the log fraction and adding a corresponding minus sign. The equation is now, clearly in the form of (6). If each likelihood is Gaussian, (Jordan et al. 1995) demonstrates that the probability of C_0 conditioned on the data becomes:

$$\begin{aligned} P(C_0|x_j) &= \frac{1}{1 + e^{w^T x_j + b}} \\ w &= \Sigma^{-1}(\mu_0 - \mu_1) \\ b &= {}^1\!/_2(\mu_1 + \mu_0)^T \Sigma^{-1}(\mu_1 - \mu_0)^T + \log(\frac{P(C_0)}{P(C_1)})) \end{aligned} \tag{4}$$

where w and b are derived in (Jordan et al. 1995) and μ_0, μ_1 are the means of the Gaussian clusters and Σ^{-1} is the inverse of the covariance matrix. This proves that the logistic function

[3]Logistic regression can also be called logistic calibration in the context of calibration related literature

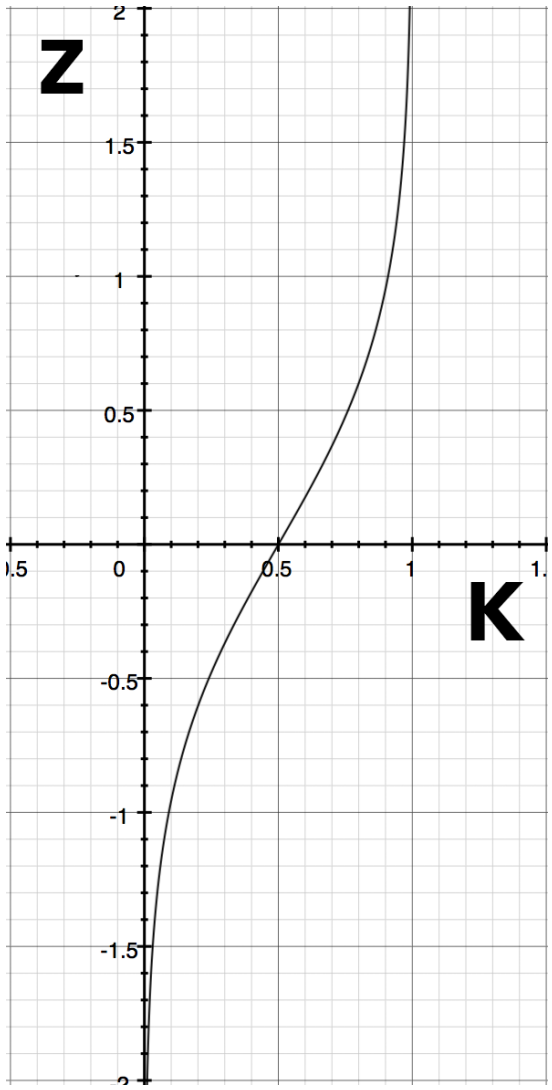


Figure 2: The logit function is of the form $Z = \dfrac{K}{1-K}$

is optimal when the probability of a class is modelled by a Gaussian and a likelihood ratio between the class likelihoods $p(x_j|C_0)/p(x_j|C_1)$ and the prior $p(C_0)/p(C_1)$.

The second perspective of the derivation of logistic regression, is sourced from (Cox 1958) which models the conditional distribution of a set of binary categorical dependent data points with a bounded function that models diminishing returns (sigmoid). The parameters of this function are then estimated by finding the inverse of the logit function (Figure 2) with respect to a linear function $f(x)$ or K with two parameters β_0 and β. These parameters are found via maximum likelihood estimation.

$$\log \frac{f(x)}{1-f(x)} = \beta_0 + \beta x \tag{5}$$

Solving for $f(x)$ and finding the inverse function, yields a sigmoid function:

$$f(x) = \frac{1}{1 + e^{-\beta_0 - x\beta}} \tag{6}$$

To determine β_0 and β, maximum likelihood is used, which involves formulating the optimization process as a binomial distribution with IID points (7).

$$L(\beta_0, \beta) = \prod_{j=1}^{N} p(x_j)^{y_j} (1 - p(x_j))^{1-y_j} \tag{7}$$

7 is solved by substituting (6) into (7), taking the logarithm of both sides, differentiating and setting the equation to zero. This then finds the most likely values of β_0 and β numerically, given the data, through gradient descent.

Paying explicit attention to the assumptions of logistic regression is important for evaluation later on. Logistic regression assumes that each data point is covered by exactly one of the n classes considered by the classifier, i.e. the closed world assumption. A consequence of this is that $P(C_0|x_j) \neq 0 \forall x_j$.

1.1.5 Softmax

The softmax function generalises logistic regression[4]. The softmax function is introduced in (Bridle 1990) and extended in (Dunne and Campbell 1997) with a vector of biases and a vector of scaling parameters for each $x_j \in C_j$, as seen in σ_1. In contemporary literature, softmax omits these scaling and bias parameters resulting in σ_0 (8), where N is the number of dimensions of the output. In the original softmax paper, (Bridle 1990), a two class neural network is constructed, such that the resulting logit vectors are Gaussian distributed in each dimension, with identical per-class variances. The author thus correctly demonstrates that softmax calibrates these logit vectors perfectly. A natural empirical question, arises which is whether the logits from neural networks, are *necessarily* Gaussian with identical per-class variances for each output dimension, given high average recall on the training data. This question is covered with an empirical analysis later on in chapter 7.

$$P(C_j|x_j) = \sigma_0 = \frac{e^{x_j}}{\sum_{k=1}^{N} e^{x_k}} \tag{8}$$

$$P(C_j|x_j) = \sigma_1 = \frac{e^{w_j x_j + b_j}}{\sum_{k=1}^{N} e^{w_k x_k + b_k}} \tag{9}$$

(Richard and Lippmann 1991), proves that minimizing cross entropy coupled with softmax as a error metric necessarily yields a *discriminatively* optimal classification function $P[d_i|X]$ *on the training data*, such that d_i is the indicator function which is one if $x \in X$ is a data instance of the ith class C_i and zero otherwise. The proof only makes guarantees regarding the

[4] A proof is provided later on

optimality of the discriminative ability for the training data only. Whether or not the network generalises to test data is currently treated solely as an empirical question in the deep learning research community.

1.1.6 Temperature scaling

The form of softmax used in contemporary neural networks is different to that used in (Dunne and Campbell 1997). In particular, softmax has both its gradient and bias fixed to unit 8. Temperature scaling, (11), is a compromise between these two forms, containing a scaling parameter τ_0, optimized on the log loss. Temperature scaling can be seen in a visual representation, later on, in Figure 3.

$$p_j = \max_N \sigma_0(x_j/\tau_0)^{(N)} \tag{10}$$

where
N : dimensionality of softmax output (number of classes).
p_j : calibrated probability estimate for input x_j
x_j : logit vector output
τ_0 : temperature scale parameter.
σ_0 : softmax function as in (8).

$$P(C_j|x_j) = \frac{e^{x_j/\tau_0}}{\sum_{k=1}^{N} e^{x_k/\tau_0}} \tag{11}$$

Temperature scaling was introduced in (Bridle 1990), but used recently in (G. Hinton, Vinyals, and Jeff Dean 2015). Here, small temperature values emphasize differences in logit vectors, a property deemed useful when transferring 'knowledge' implicit in subtle logit differences from a cumbersome knowledgeable large network to a small nerwork potentially placed on mobile devices. In the context of literature on adversarial attacks, temperature scaling is used as a defence, in defensive distillation (Papernot, McDaniel, Wu, et al. 2016), which attempts to remove adversarial attacks with pre-determined temperature values, by lowering the magnitude of the gradients. This is the first link I discovered between methods used for calibration and methods used for adversarial defences.

(Niculescu-Mizil and Caruana 2005) demonstrates in 2005, that artificial neural networks predict well calibrated probabilities, but (Guo et al. 2017) finds that the gap between calibration and accuracy has diverged recently as the depth of neural networks have grown. In order to reduce the recent increase in calibration loss for deep neural networks, (Guo et al. 2017) suggests temperature scaling (11). Here, the discriminative ability of the back-propagation algorithm accounts for the decision boundary. Temperature scaling acts as a decision boundary method. Figure 3 shows the effect of temperature scaling on Gaussian

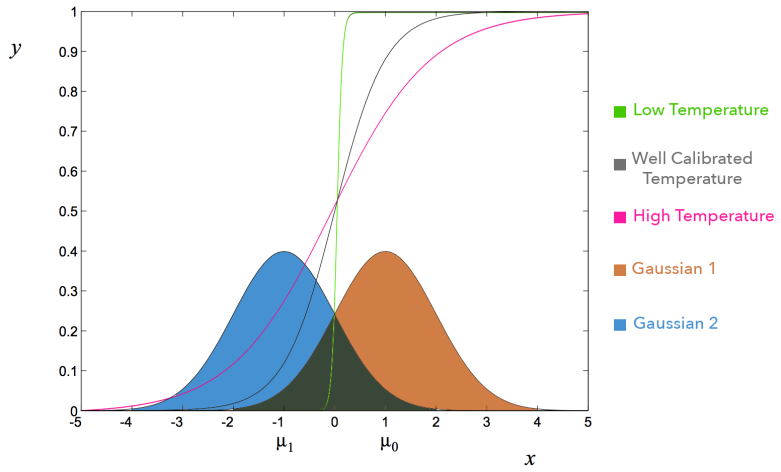


Figure 3: Various temperature values for P(Gaussian 1| x). The x-axis represents model score values and the y-axis is the associated probability value. The class probabilities at each point are $fpos/(fpos + fneg)$, where $fpos$ & $fneg$ are the per class conditional density values. In high dimensions, each class represented by the sigmoid function will include exponentially more area, due to the curse of dimensionality.

clusters for temperature values that are either too high, too low or correct temperature values τ_0. The reason temperature scaling is used for calibration is that the temperature modifies the gradient of each Gaussian with the optimized constant τ_0, such that each point on the sigmoid is more indicative of the likelihood ratio between classes. The effect of this is as $\tau_0 \longrightarrow \infty$, the probability of each class $\longrightarrow 1/n$ where n is the number of classes, and as $\tau_0 \longrightarrow 0$, the probability of the highest class $\longrightarrow 1$ whilst the other classes tend to $\longrightarrow 0$. A clear issue can be seen almost immediately with this method, more specifically, if each per-class distribution is Gaussian with different variances from each other, then no constant τ_0 can calibrate the Gaussians perfectly.

1.2 Beta calibration

Beta calibration (Kull, Silva Filho, Flach, et al. 2017) is based on the beta distribution which includes many notable functions such as the logit, sigmoid and identity. This allows it to calibrate scores produced by models such as naive bayes, which biases its scores towards extremities when the assumption of feature independence is not met, using the inverse sigmoid or logit function, a function which is obviously not in softmax's repertoire [see Fig. (2)]. If models produce well calibrated probabilities a priori and need no calibration, then, as softmax does not include the identity map, it is guaranteed to mis-calibrate these probability estimates.

Figure 4 shows a 3x3 matrix revealing the expressive power of the beta calibration procedure for different parameters a,b and midpoint m[5]. These parameters are arrived at through the

[5]There exists an additional optional parameter c imbuing beta calibration with additional expressive power, that is not mentioned here

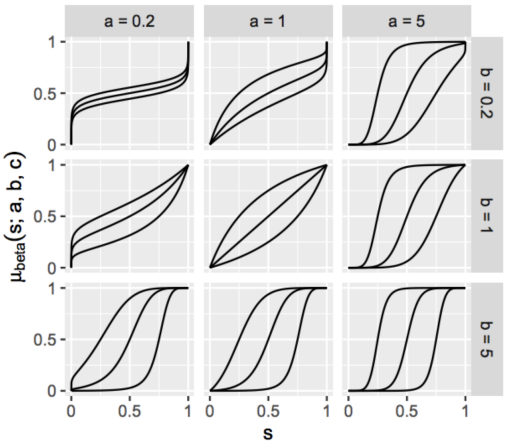


Figure 4: Calibration maps for the beta distribution. Source: (Kull, Silva Filho, Flach, et al. 2017)

beta calibration algorithm, which, in attempting to construct a map from scores, $s \in [0, 1]$ to probabilities, optimizes for some combination of these parameters. The 3 different curves in each entry of the matrix, are arrived at for $m \in \{0.25, 0.5, 0.75\}$. The top left entry shows the inverse sigmoid and the bottom right shows sigmoid shapes, which could be used to calibrate scores from Naive Bayes and SVM models, respectively.

1.3 Beta calibrations relationship to temperature scaling

As part of this thesis, a short proof is demonstrated, showing the relationship between the 2-parameter beta calibration algorithm and temperature scaling.

Lemma 1.1: *Softmax (σ_1) is a generalization of the logistic (sigmoid) function.*

The sigmoid function is calculated as follows:

$$\begin{aligned} P(C = 1|x) &= \frac{1}{1 + e^{w_1' x + b}} \\ P(C = 2|x) &= 1 - P(C = 1|x) \\ &= \frac{1 + e^{w_1' x + b}}{1 + e^{w_1' x + b}} - P(C = 1|x) \\ &= \frac{e^{w_1' x + b}}{1 + e^{w_1' x + b}} \end{aligned} \tag{12}$$

Softmax (σ_1) in the two class case is:

$$
\begin{aligned}
P(C=1|x) &= \frac{e^{w_1 x+b_1}}{e^{w_2 x+b_2}+e^{w_1 x+b_1}} \\
&= \frac{e^{w_1 x+b_1}}{e^{w_2 x+b_2}+e^{w_1 x+b_1}} \times \frac{e^{-w_1 x-b_1}}{e^{-w_1 x-b_1}} \\
&= \frac{1}{1+e^{w_2 x-w1 x+b_2-b_1}} \\
P(C=2|x) &= \frac{e^{w_2 x+b_2}}{e^{w_2 x+b_2}+e^{w_1 x+b_1}} \\
&= \frac{e^{w_2 x-w_1 x+b_2-b_1}}{e^{w_2 x-w_1 x+b_2-b_1}+1}
\end{aligned} \tag{13}
$$

The two class softmax function (σ_1) is therefore equal to the sigmoid function if $w_2 - w_1 = w_1'$ and $b_2 - b_1 = b$. Temperature scaling uses a fixed gradient τ_0 with the softmax function σ_1 and a vector of zero initialised biases. N.B. $\tau = \frac{1}{\tau_0}$.

$$
\begin{aligned}
P(C_j|x_j) &= \frac{e^{x_j/\tau_0}}{\sum_{k=1}^{N} e^{x_k/\tau_0}} \\
&= \frac{e^{\tau x_j}}{\sum_{k=1}^{N} e^{\tau x_k}} \\
&= \frac{1}{\sum_{k=1}^{N} e^{\tau x_k - \tau x_j}}
\end{aligned} \tag{14}
$$

This clearly shows that temperature scaling is softmax with a fixed weight τ. This is clearly true since $\forall w_i \mapsto \tau$. The relationships between these functions are important to cover so in the analysis section later, the dependencies and weaknesses between the functions can easily be seen.

Lemma 1.2: *Temperature scaling is not a strict subset of beta calibration, for $\tau \neq 0$ with score vectors of dimension at least 2.*

The proof of this lemma assumes that the beta calibration method does not use the softmax function during calibration, in contrast to temperature scaling, which is assumed to inherently use the multi-dimensional softmax function mentioned above. This proof is completed with the use of a counter example. It is first important to note a few facts that are proven in (Kull, Silva Filho, Flach, et al. 2017). The paper also usefully proves (15), (16), (17)

$$
\mu_{beta}(s; a, a, c) = \mu_{logistic}(\ln(\frac{s}{1-s}); a, c) \tag{15}
$$

$$
\mu_{logistic}(s; a, c) = \frac{1}{1+\frac{1}{e^{as+c}}} \tag{16}
$$

$$
\mu_{beta}(s; a, a, c) = \frac{1}{1+\frac{1}{e^c \frac{s^a}{(1-s)^a}}} \tag{17}
$$

(17) can then be transformed through simple algebraic manipulations into (18)

$$\mu_{beta}(s; a, a, c) = \frac{1}{e^{-c}(1-s)^a s^{-a} + 1} \tag{18}$$

(18) is the mapping performed by s_{test} in algorithm 1 in (Kull, Silva Filho, Flach, et al. 2017).

The first important step in this proof consists of generating example scores for two data points with an arbitrarily chosen temperature value t. The resulting probability estimates from piping the scores through the temperature scaling function will reveal that *no values of a and c*, in the context of beta calibration, can, *in the general case*, re-create the probability estimates that the temperature scaling function produces.

Let $\{s_1, s_2, s_3, ...s_n\} = \{1/4, 3/4, 0, ..., 0\}$ and let $t \in \mathbb{R} = \tau$. Piping $\{s_1, s_2\}$ through temperature scaling T_s (14), yields (19), s.t. $s_i \sim q_i$.

$$\{T_s(s_1), T_s(s_2)\} = \{q_1, q_2\} = \{\frac{1}{1 + e^{t/2} + \sum_{i=2}^{n} e^{-t/4}}, \frac{1}{1 + e^{-t/2} + \sum_{i=2}^{n} e^{-3t/4}}\} \tag{19}$$

Temperature scaled probability estimates will now be equated to beta calibration probability estimates in order to derive a contradiction through variable substitution. The first observation that needs to be made is merely structural, in particular that the scores from temperature scaling and beta calibration can be equated in the following way.

$$\frac{1}{e^{-c}(1-s)^a s^{-a} + 1} = \frac{1}{1 + e^x} \tag{20}$$

$$1 + e^x = e^{-c}(1-s)^a s^{-a} + 1 \tag{21}$$

$$x = -c + a(ln(1-s) - \ln(s)) \tag{22}$$

The above equations only hold if both denominators of q_1, q_2 are of the form $1 + e^x$. From this deduction, it is clear that the formulas in 23 need to hold if there exist values of c and a that equate beta calibration to temperature scaling.

$$\begin{aligned} x_1 &= -c + a(\ln(1-s) - \ln(s)) \\ -c &= x_1 - a(\ln(1-s) - \ln(s)) \\ x_2 &= x_1 - a(\ln(1-s) - \ln(s)) + a(\ln(1-s) - \ln(s)) \\ x_2 &= x_1 \end{aligned} \tag{23}$$

This means that if both denominators of q_1, q_2 are of the form $1 + e^x$ then it is simply valid to equate arbitrary $x_1 \wedge x_2$ and solve to find the valid values of a and c.

$$\begin{aligned} q_1 &= \frac{1}{1 + e^{t/2} + e^{-t/4}(n-2)} \\ q_2 &= \frac{1}{1 + e^{-t/2} + e^{-3t/4}(n-2)} \end{aligned} \tag{24}$$

$$
\begin{aligned}
q_1 &= \frac{1}{1 + e^{t/4}(e^{t/4} + e^{-t/2}(n-2))} \\
q_2 &= \frac{1}{1 + e^{-t/4}(e^{-t/4} + e^{-t/2}(n-2))}
\end{aligned}
\tag{25}
$$

By equating the formulas together, it should reveal which values of t for which values of a and c can also be found.

$$
\begin{aligned}
e^{t/4}(e^{t/4} + e^{-t/2}(n-2)) &= e^{-t/4}(e^{-t/4} + e^{-t/2}(n-2)) \\
t(e^{t/4} + e^{-t/2}(n-2)) &= -t(e^{-t/4} + e^{-t/2}(n-2)) \\
te^{t/4} &= -te^{-t/4} \\
t(e^{t/4} + e^{-t/4}) &= 0
\end{aligned}
\tag{26}
$$

The last line on (26) has no real solutions for $t \neq 0$ $\square$.

This result proves that in general, no value of a and c exist for $t \neq 0$, for scores with at least two dimensions. I also speculate that similar results could be proved for the 3-parameter beta calibration algorithm.

1.4 Conclusions

Different calibration methods have been discussed in this section, each with different strengths and weaknesses. Logistic regression, which uses the assumption of per-class Gaussian distributed data in calibration, through to beta calibration, which adds greater expressive power, by including calibrating maps such as the inverse sigmoid. A proof has also been demonstrated between temperature scaling and beta calibration. A question that arises, is, can these calibration methods help in dealing with adversarial attacks on neural networks, that undermine the probability estimates from classifiers.

2 Adversarial attacks

The calibration procedures mentioned in chapter 1 have not been used in deep learning/calibration literature to calibrate adversarial. Adversarial examples are specially crafted instances generated by so called adversarial attacks. These attacks generate, or manipulate, existing data, mostly images, to achieve poor performance when classified by neural networks. This chapter discusses recent adversarial attacks, to give context to the defences proposed in the next chapter, and thus the new defence that I propose later on in chapter 5.

Adversarial attacks reliably craft data that consistently scores poorly on metrics of calibration and have only found themselves in the literature very recently: (Nguyen, Yosinski, and Clune 2015), (I. J. Goodfellow, Shlens, and Szegedy 2014), (Dong et al. 2018) & (Carlini and Wagner 2016), (Gu and Rigazio 2014), (Szegedy et al. 2013). Adversarial attacks also generalize to printed real world photographs (Sharif et al. 2016), which raises important philosophical questions about the use of neural networks in the real world. Adversarial examples that transfer arbitrarily from one network trained on a specific dataset to a different network with different training data have also been established (Kurakin, I. Goodfellow, and Bengio 2016).

2.1 Types of adversarial attack

Adversarial attacks can have varying access to the neural network's underlying architecture:

1. white box attacks - the attack method has full knowledge of all model parameters from the number of layers to the weight values.
2. black box attacks - the attack method only has access to the predicted class or the confidence scores.

The examples created by the attacks can be instances of:

1. false positives - e.g. a benign program classified as malware.
2. false negatives - e.g. a malware classified as benign.
3. nonsense images - e.g. white noise that is classified as a class with high confidence.
4. clear cut images - e.g. a clear image of a cat that is classified as a road sign.
5. targeted attacks - e.g. the instruction to create a false positive of a particular class.
6. non-targeted attacks - e.g. the instruction to create a false positive of any class other than its own.

In the context of adversarial attacks, it is commonplace to view the network as the map:

$$F : X \mapsto Y \tag{27}$$

where X is a vector of raw features and Y is an output vector. Adversarial attacks then solve the optimisation problem in (28).

$$\arg\min_{\delta_x} ||\delta_x|| s.t. F(X + \delta_x) = Y^* \tag{28}$$

An adversarial example X^* is constructed by adding a perturbation vector δ_x, where Y^* is the desired adversarial output and $X + \delta_x \in [0, 1]$ where $[0, 1]$ is the upper and lower bound of a well formed example, subject to the constraints in 28. Intuitively, these constraints coerce the network into mis-classifying each example with a minimal perturbation vector.

The distance metrics that measure the size of the perturbation are normally chosen from the following list.

1. $\mathcal{L}_p$ metrics (29), such that $\mathcal{L}_0$ is the number of changed pixels, $\mathcal{L}_2$ is the euclidean distance between the two images and $\mathcal{L}_\infty$ is the maximum change of any of the pixels.

$$||x||_p = (\sum_{i=1}^{N} ||x_i||^p)^{1/p} \tag{29}$$

2. PASS score (Rozsa, Rudd, and Boult 2016) is a metric designed to better reflect notions of psycho-physical similarity than $\mathcal{L}_p$ metrics.

2.2 Adversarial attack methods

2.2.1 L-BFGS

This attack was the first attack method proposed for neural networks by (Szegedy et al. 2013). It uses the idea of optimisation on the smoothness prior. The smoothness prior states that images within a ball of radius ϵ, of a training example should, with high probability, be assigned the correct class. This idea was then optimised against using linear trial and error to find a c for each image such that c multiplied by an arbitrary small perturbation vector ϵ mis-classifies the image.

2.2.2 FGSM

FGSM (I. J. Goodfellow, Shlens, and Szegedy 2014) builds on the L-BFGS method, replacing expensive linear search with back-propagation to find the perturbation vector, performing gradient ascent with respect to the correct class. FGSM adds a weak noise vector, only adding more noise if the error increases i.e. the derivative of the cost function with respect to the parameters increases. (Dong et al. 2018) adds momentum to escape local minima in the error landscape. Targeted attacks can be easily formulated with FGSM.

FGSM uses the optimisation strategy in 30 to create the perturbation vector η, such that x is the input to the model, y is the predicted class, θ are the model parameters and $J(\theta, x, y)$ is the cost function. ∇_x indicates the derivative of the cost function is taken with respect to the

input x. The sign function takes the sign of the resulting derivative. The adversarial example is then constructed as $x' = x + \eta$, which means if the derivative is negative, then the adversarial perturbation will move the example away from the current point and vice versa if the derivative is positive.

$$\eta = \epsilon sign(\nabla_x, J(\theta, x, y)) \tag{30}$$

In (31), the perturbation is labelled as ϵ. The difference of notation is consistent with (I. J. Goodfellow, Shlens, and Szegedy 2014)

$$\epsilon * w^T + \epsilon_{noise} * w^T = \epsilon * w^T \tag{31}$$

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{10000} \\ x_{21} & x_{22} & x_{23} & \dots & x_{20000} \\ \dots & \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & x_{n3} & \dots & x_{n100000} \end{bmatrix} \begin{bmatrix} \epsilon_{11} \\ \epsilon_{21} \\ \dots \\ \epsilon_{100000} \end{bmatrix} = \begin{bmatrix} \epsilon_{11} * x_{11} + \epsilon_{12} * x_{12} + ...\epsilon_{1n} * x_{10000} \\ \epsilon_{21} * x_{11} + \epsilon_{12} * x_{12} + ...\epsilon_{1n} * x_{10000} \\ \dots \\ \epsilon_{n1} * x_{11} + \epsilon_{12} * x_{12} + ...\epsilon_{1n} * x_{10000} \end{bmatrix}$$

Neural networks with large number of linear ReLu activation functions are especially vulnerable to FGSM attacks as the activation grows in proportion to ϵnw, where n is the dimension of the weight vector, w is the average weight value and ϵ is the perturbation amount. This can be seen in (30). Non-linear sigmoidal activation functions have been found to be harder to optimise, yet still had linear behaviour that could be exploited. In particular, small perturbations yield large output differences if the network has many weights greater than 1. Given a modest neural network with 1,000,000 weights, any column vector multiplied with it (feeding an image into the network), with only small ϵ_{noise} can end up very far away in the output space, especially if the output is of high dimension.

2.2.3 Basic Iterative Attack

The Basic Iterative Method (Kurakin, I. Goodfellow, and Bengio 2016) is an extension of FGSM, that uses a smaller noise vector ϵ but repeats a attack identical to FGSM iteratively with a variable α which scales the FGSM noise vector before applying a clip operation to the resulting image, keeping it within the bounds of maximum image pixels after each iteration.

2.2.4 JSMA

JSMA is a forward derivative approach, which works differently to gradient descent based approaches. JSMA uses a Jacobian matrix (32), which simply is a large matrix with each column representing the first order derivatives for every pixel with respect to the classification. This resulting matrix is known as an adversarial saliency map, that marks the points at which the highest change in classification is produced for each pixel. The pixel corresponding to the maximal entry in the forward derivative or Jacobian matrix is changed, for a number of pixels

that is parameterised. Low values for the forward derivative indicate regions of the instance space which are unlikely to yield adversarial examples.

$$J_F(x) = \frac{\partial F(x)}{\partial x} \tag{32}$$

where F is a neural network, x is an input image, F_j is the output of the neural network for a particular class.

2.2.5 DeepFool

DeepFool (Moosavi Dezfooli, Fawzi, and Frossard 2016) finds the closest distance to the decision boundary from the proposed example, to another target class which isn't the source class. It treats multiclass classifiers as combinations of binary affine classifiers.

2.3 Conclusions

Attacks occur from many different angles. It is now important to cover how these attacks are mitigated. The literature generally engages in a cat and mouse struggle, whereby an attack is created, after which a defence to combat that attack is also created and vice versa. The next section covers some of these defences.

3 Defences against adversarial attacks

The main contribution this thesis brings, is a calibration based adversarial defence strategy. As such, it is important to include similar attempts to defend against adversarial attacks. This section will give a brief overview of different types of adversarial defence mechanisms, which have only emerged in the literature recently. Defences can be either or both of the following.

1. *Proactive models* : predict and prevent attacks before they happen.
2. *Reactive models* : focus on responding to attacks as and when they come up.

Defences can also have various strengths in dealing with data.

1. *Adversarial detecting* : Detect individual examples as adversarial.
2. *Adversarial correcting* : Detect individual examples as adversarial and re-classify them into their correct class.

3.1 Proactive defences

3.1.1 Network distillation

A link to calibration and adversarial attacks, arises here. In the previous section, temperature scaling was used to adjust the gradient of the multi-dimensional sigmoid function and also in knowledge transferal between networks. In the context of defences against adversarial attacks (Papernot, McDaniel, Wu, et al. 2016) uses temperature scaling with fixed temperature values to manipulate the ease with which gradient based methods find the gradient. Defensive distillation changes the success rate of an adversarial attack from 90% to 0.5% when high temperature values are used.

3.1.2 Why does defensive distillation work?

Defensive distillation changes the gradient of the softmax function for examples fed into the secondary network used to classify examples, making the gradients very small. This is achieved by using a very large temperature value meaning the values approach $1/n$. (Papernot, McDaniel, Wu, et al. 2016) only tests their method against the JSMA attack, which when empirically tested in this paper, consistently use smaller logit differences than the training and test logits, but still large enough to produce high confidence values in the two class case.

3.2 Reactive defences

3.2.1 Neural network training as a defence

(Metzen et al. 2017) uses an additional network as a sub-unit of the main network that acts as a binary classifier detecting whether an image is adversarial or not. Figure 5 shows the accuracy of this method. Defences using neural networks were introduced in (I. J. Goodfellow, Shlens, and Szegedy 2014), whereby adversarial examples are used in the training procedure.

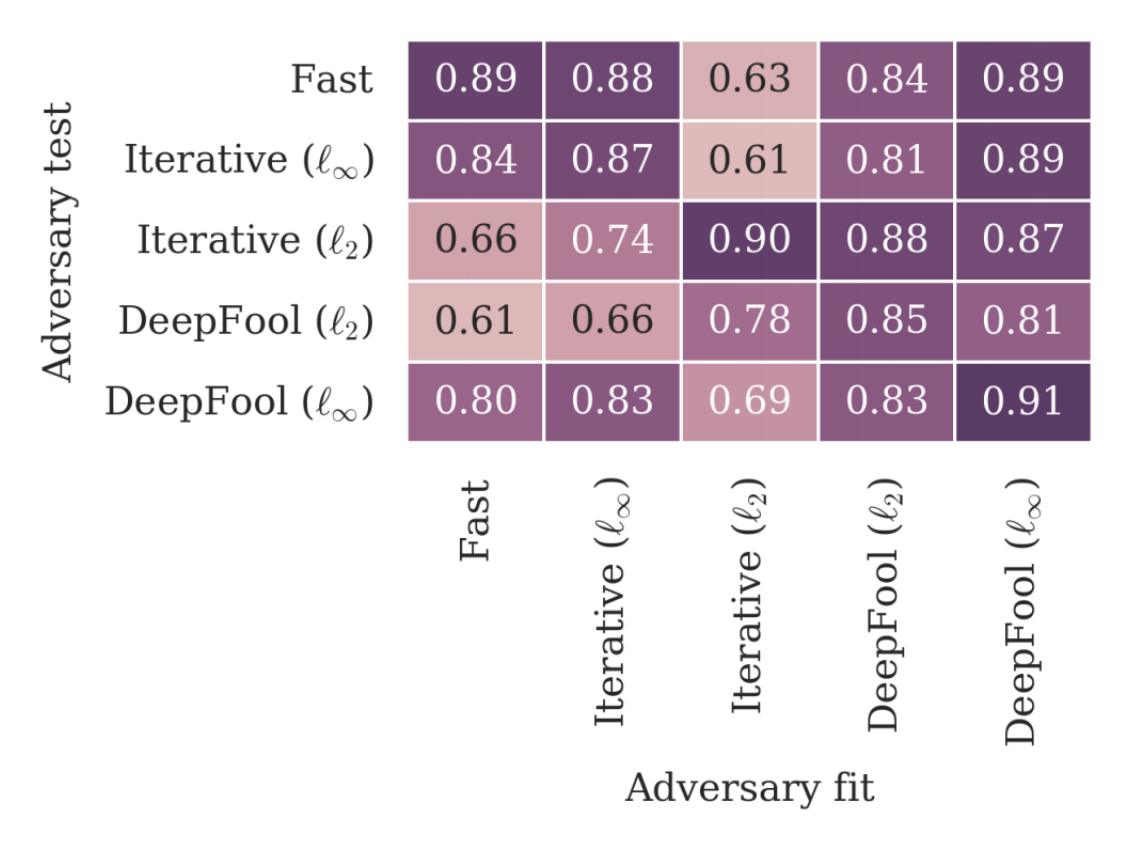


Figure 5: The error rate on adversarial examples achieved by (Metzen et al. 2017) on the CIFAR10 dataset.

In this method, adversarial images of the class A, which pretend to be class B, are placed in the training set of class A. The accuracy of this method can be seen against ϵ in Figure 6.

The obvious disadvantage of this method compared to the method proposed in this thesis is that often the network cannot be explicitly trained on all attack types and parameters, resulting in a degradation of the generalisation (from training to test data) performance. Figure 5 shows that in all cases, when adversary test data is used which does not match the adversarial training fit the accuracy always reduces.

3.2.2 Statistical defences

(Grosse et al. 2017) trains their network on specific attack types and parameters with an additional outlier class for adversarial examples. They then use a multi-dimensional statistical test over the maximum mean discrepancy and the energy distance on *input* features to classify instances as adversarial. The paper claims that input features for adversarial instances are far outside of the test and training distributions using the maximum mean discrepancy and energy distance. Intuitively this method finds input features that characterise natural images. Other work has been done by (Bradshaw, Matthews, and Ghahramani 2017), who use Gaussian

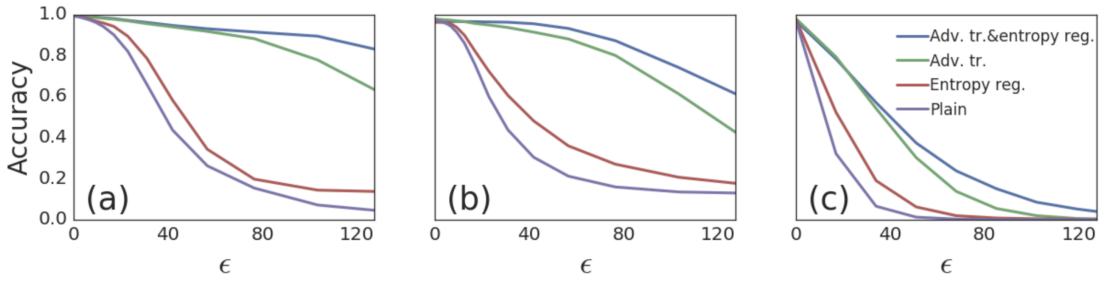


Figure 6: a) and b) show epsilon values for the FGSM attack versus accuracy on the MNIST dataset. One can see that with adversarial traning, the accuracy is higher for all values of *epsilon* than without it. c) shows a new attack called PGD. Source: ICLR 2018 openreview.net - Intriguing properties of adversarial examples. Interestingly, this method has the opposite performance with increasing values of ϵ.

Processes on top of a conventional convolutional neural network architecture, with a radial basis kernel to imbue the network with a way of understanding its own perceptual limits. The authors find that the network becomes more resistant to adversarial attack.

3.3 Conclusions

Having reviewed the defences, I will now propose an original defence against adversarial attacks, using principles from the literature on calibration. The most similar method to the defence proposed in the next section is (Grosse et al. 2017), as this uses purely distributional approaches to rooting out the adversarial attacks. The main difference though, is that (Grosse et al. 2017) uses a measure over the *input* features rather than the output features, which is the approach taken here. Having reviewed the literature, I can conclude safely that the methodology proposed in the next section will be entirely new content for the literature regarding adversarial defences.

4 Background Check

Background Check (Perello-Nieto et al. 2016), is the calibrating procedure that this thesis uses to defend neural networks from adversarial attacks. Background Check takes as input, the scores from logit vectors and maps them to probabilities, replacing softmax *after training*. Background Check provides a probabilistic framework for the classification of regions as outliers or not. These outlier regions will be classified as adversarial regions (regions of space where adversarial attacks find their examples), in the context of a world where the promise is made, that test instances are either adversarial or valid test data. This promise can be assumed within the context of this thesis. Valid test data contains at least one grounded consistent representation within each data point, in line with the training data.

Background Check also helps with a fundamental issue of neural networks that have a final softmax layer. The issue, is that of the impossibility for all elements of the post-softmax output vector to simultaneously achieve probabilities below $1/n$, where n is the number of output dimensions. In practice, elements of the post-softmax output vector never have exactly the same value and are always different by at least an arbitrarily small ϵ. This means, that arbitrary data, fed into a neural network, almost always has an arbitrary classification. Background Check helps by providing a way to check if the example is far away from the training data density and henceforth classify the example as uncertain. Labelling regions with an outlier class may be pragmatic and safe if the dimensionality of the classification function subsumes large swathes of space, such that the reverse output to input mapping is unclear. In the case of 32×32 images from CIFAR10, there exist 16.5×10^6 or 255^3 different values for each coloured pixel and 1024 pixels in total. This combination implies $(16.5 \times 10^6)^{1024}$ different possible images. Compared to fourty-five thousand training images that accompany CIFAR-10, the relative proportion of training data, compared to the possible number of images of that size, is less than an atom in the observable universe (10^{82}). There are only ten object classes to represent all images. As such, it seems likely that many output points may represent classifications from points in space that have seen very little training data. This combinatoric analysis may provide some intuition as to why Background Check may be effective in this context.

Background Check also provides a framework to resolve ambiguity inherent in a single value representing probability. More specifically, Background Check provides two values to represent a single probability value of a data instance. One value represents distance from the data density, and another represents the certainty of a particular class. This approach avoids overloading the meaning of a single number representing probability. More specifically, an uncertainty of $1/n$ for all classes, could represent an instance very close to the decision boundary or very far away from training data. On the other hand, an output of zero, could represent a point very far away from training data or a classification that the data is definitely not that particular class.

4.1 How does Background Check work?

Background Check has three explicit goals:

1. Perform cautious classification with a reject option.
2. Identification of outliers.
3. Better assess confidence in predictions.

In this thesis, all three goals are exploited, but the most focus is on the second usage. The first key principle that Background Check uses is the introduction of an additional outlier background class, b. b represents regions of space where data is sparse or non-existent. Then, a foreground class is introduced, f. This represents regions of space where data is plentiful or dense, i.e. data from any class apart from the background class, is abundant. b is introduced as an additional class whilst f is kept as a reference class. Every instance x necessarily belongs to either f or b. $P(b|x) = 0$ and $P(f|x) = 1$ refers to absolute certainty that the instance belongs to one of our classes with *sufficient* training data, where $P(C|x)$ is a conditional probability measure. The ratio of the two conditional measures defines the reliability factor $r(x)$. $r(x)$ is the second probability value describing how far the data is away from the density.

$$r(x) = \frac{P(f|x)}{P(b|x)} \quad = \frac{P(f_1|x) + ... + P(f_n|x)}{P(b|x)} = \frac{P(f,x)P(x)}{P(b,x)P(x)} = \frac{P(x,f)}{P(x,b)} \tag{33}$$

If $r(x) \leq 1$ in (33), the classification that the reliability factor indicates is b, else if $r(x) > 1$ then the classification is f. $P(x, f)$ and $P(x, b)$ are referred to as the foreground and background densities. Furthermore, the relative foreground and background densities can be defined, $q_f(x)$ and $q_b(x)$.

$$q_f(x) = \frac{P(x,f)}{\max_x P(x,f)}, \qquad q_b(x) = \frac{P(x,b)}{\max_x P(x,f)} \tag{34}$$

Intuitively, the relative density outputs the proportion of f or b, at the point in space corresponding to the instance x being evaluated. Simple dividing $q_f(x)$ by $q_b(x)$ yields $r(x)$.

4.1.1 Justification of $q_b(x)$

To construct $q_b(x)$, four inductive biases, in increasing strength, are given. This thesis only uses the third inductive bias.

1. *Inductive bias 1* : $q_b(x)$ is a function of $q_f(x)$. This is justified, by the idea that with no other information, there is no reason to assign different background densities to points with the same foreground density. The domain knowledge informs the function used $\mu : [0, 1] \longrightarrow [0, \infty)$.
2. *Inductive bias 2* : monotonicity of μ, that is, when moving to a region with higher foreground density the background increases or decreases.

3. *Inductive bias 3* : an affine bias, i.e. $\mu(x) = ax + b$ or by replacing a and b: $\mu(x) = (1 - x)\mu(0) + x\mu(1)$.

4. *Inductive bias 4* : constant background ie $\mu(0), \mu(1) = 0.5$.

4.1.2 Performance of Background Check

The key notion in Background Check is the implementation of the inductive bias that shapes q_b. This implementation is provided in two different different ways.

1. *BCD* : referred to as the discriminative approach, involves generating artificial background instances around foreground data and then training a binary discriminative classifier to separate them. The instances are generated in a hypercube or a hypersphere, such that the background is half as dense as $max_x P(x, f)$.

2. *BCF* : referred to as the familiarity approach, this involves fitting a one-class model on the foreground data to get q_f, then using an inductive bias to obtain q_b. The data, x, that is being fit must obviously contain an underlying measure. The familiarity factor $r(x)$ can then be found, and from that, the posterior probabilities $P(b|x)$ and $P(f|x)$ can be computed.

4.2 Conclusions

Background check crucially provides a framework for the identification of outliers. The reliability factor can be used to find these outliers once both the foreground and background relative densities have been established. This is an important concept in considering the next section, where Background Check will be employed to root out adversarial examples.

5 Methodology

5.1 How is Background Check used?

Background Check is used in this thesis, to model the reliability factor that classifies images as adversarial or not. First, the decision was made to use the BCF approach due to its simplicity and speed, especially in higher dimensional spaces. The measure underlying the space of data was the one-dimensional L_1 difference between elements of the logit vector. For instance, if a neural network outputs a vector $[-5, 5]$ where the first element corresponds to the score of a dog, and the second to a plane, then the score difference is 10. This measure only works for neural networks that produce two dimensional logit vectors as output.

The reason this measure was used, was because the magnitude of this difference dictates the distance of an instance to the softmax decision boundary. This can be seen clearly given an example. If one output neuron assumes a value that approaches ∞ and another output neuron assumes a value approaching -∞, then in the limit, the softmax probability of the first positively valued neuron, tends to one and the other negatively valued neuron, has a softmax probability that tends to zero. If both neurons have an equivalent value, then softmax assumes a value of 0.5, for both neurons. This metric was used because I noticed that adversarial attacks seemed to score extremely high confidence scores on images. Perhaps, the diminishing effect of softmax was hiding large differences in logit space when researchers were observing their results. In this space, even small score differences as small as 6 imbue the positive neuron with an extremely high confidence in softmax space. For instance, a two-class neural network with one neuron assigned a value of 3 coupled with a second neuron with value -3, results in a softmax probability of the positively valued neuron to be 99...%. As such, it was computationally simpler to use the L_1 difference to separate adversarial instances from test and training instances.

The one-class model used to fit $q_f(x)$ was a gamma function, with the parameters determining the shape of the function decided by scipys gamma.fit procedure, which uses a maximum likelihood estimation on the three shape parameters of a gamma function. The gamma function was fit on lists of *training data* in logit space. To establish the link from $q_f(x)$ to $q_b(x)$, the third inductive bias was used with $\mu(0) = 1$ and $\mu(1) = 0$. Using domain knowledge, the power of this function was raised to five, justified by a few glimpses at some of the initial data. The specific equation used can be seen in (35).

$$q_b = (1 - q_f(x))^5 \times \mu(0) + q_f(x) \times \mu(1) \tag{35}$$

During classification of test data, the value of $r(x)$ was computed to detect the distance to the data density. If it is greater than 1 then whichever class the neural network claimed it to be was assigned as its class. If, on the other hand $r(x)$ was less than 1 then it was assigned the adversarial class. The resulting form of Background Check, can be seen in Figure 7.
In order to assess the structure of the logit difference distributions to find where adversarial

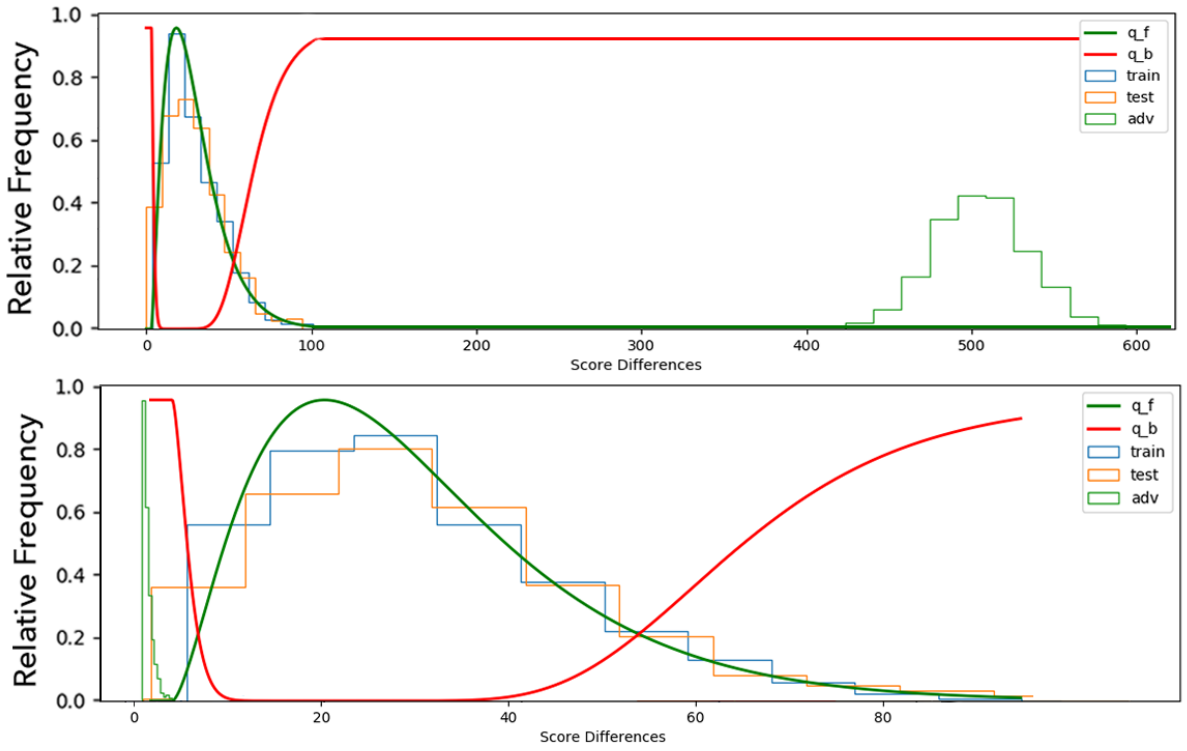


Figure 7: Background Check applied to the score difference of two, two class neural network logit vectors (one network each for the top and bottom figures). The training, test and adversarial images, in blue, orange and green, respectively, are organised into ten bins each. The adversarial data for the top figure was generated by the momentum attack method (Dong et al. 2018), with very large parameter settings. The bottom figure has adversarial data crafted by the JSMA attack (Papernot, McDaniel, Jha, et al. 2016) with typical parameter settings. The green line represents q_f, the foreground relative density and q_b represents the background relative density. It is clear, that the adversarial images in both cases are distinctly separated from the training and test data, which overlap to the point of indistinguishability. In the top figure, the adversarial images have logit differences much larger than the training/test data compared to much smaller logit differences for the bottom figure. Whilst the bottom figure has logit differences around 0.5 to 2, this can still produce class probabilities over 70%.

examples sit in logit space, one neural network for each attack type, parameter combination and dataset, was trained on a Nvidia P100 GPU. This neural network was tasked with classifying a variety of paired class combinations from the CIFAR10 dataset. The code to build the network was forked from the footnoted Github link[6] and then extensively built on. The network had regularising techniques appended to it such as L2 regularisation and dropout. Seven different adversarial attacks were tested against the networks with a variety of different parameters guiding the attacks, for each attack. The attacks were all white-box attacks and performed on the network which included a final softmax layer in its structure. The data analysis and application of Background Check was performed on the resulting logit vectors in over one thousand lines of original Python based code. The code can be found on my GitHub under the Thesis section[7] and is modular and readable. The GitHub includes the logit vectors used for the data analysis and the code for the neural network. The final two-class average recall of each network on the validation set of the network was always above 80% after only 300 iterations over the training data. All attacks were implemented in Python using the cleverhans API (Papernot, Carlini, et al. 2016) and TensorFlow (Abadi et al. 2016).

In order to obtain the logit vector of a particular image x, x was fed in a feed-forward manner to the trained network. Class combinations used during training and testing were sampled from CIFAR10. These pairs were: dog versus plane (DVP), fish vs ship (FVS) and airplane vs horse (AVH). Different paired combinations of CIFAR10 were deployed to ensure that the phenomenona regarding the placement of the test, training and adversarial distributions observed, were not simply a function of one specific individual data set. In addition, the network was re-trained for all attack types with different random seeds. However, clearly testing multiple networks on three different paired combinations does not guarantee that this is a byproduct of the network type used or the CIFAR10 dataset, it just makes it more unlikely, similar to bootstrap aggregation. The constructed network was moderately simple and shallow with the wiring, activation functions and convolution kernel sizes are described in the Neural Network Architecture Table.

[6] https://github.com/COMSM0018-Applied-Deep-Learning/labsheets

[7] https://github.com/oliverjgoldstein/Thesis

5.2 Architecture

Neural Network Architecture					
Layer #	Units	Layer Type	Kernel	Activation	Stride
1	32	Convolutional	5x5	ReLu	1
2	n/a	Pooling	2x2	Max	2
3	64	Convolutional	5x5	ReLu	1
4	n/a	Pooling	2x2	Max	2
5	256	Fully Connected	n/a	ReLu	n/a
6	2	Fully Connected	n/a	Linear	n/a

Batch Size	Learning Rate	Optimiser Type	Bias Init	Weight Init
256	0.001	Adam	0	Uniform(0,1)

6 Results

6.1 Visualisation of results

The following three categories were established over the L_1 logit difference space. The second category is such that Background Check will find it harder to separate adversarial images from test images. To see all histograms please refer to the appendix.

1. Adversarial examples in a distinct cluster, closer to the decision boundary, than the training and test data.

2. Adversarial examples scattered amongst the training/test data.

3. Adversarial examples in a distinct cluster, further from the decision boundary, than the training and test data.

6.2 Description of images

Figures (8 - 13) display an example adversarial image in the centre of each image, between the two matrices. These central images are sourced directly from the specifc attack corresponding to a DVP CIFAR-10 pair. Some of these adversarial images contain colored noise as a result of large perturbation vectors applied to them, though I argue, that these images should not be ignored as much as they are in mainstream literature, as these images can occur in the real world and consequently, be just as damaging as adversarial images with small perturbation vectors. In all of the figures, if a dog is shown in this central picture, the classifier would believe, that image to be, with high confidence a plane and vice versa. The top left confusion matrix is that of the neural network without Background Check. The top right confusion matrix is that of the neural network with Background Check applied to it. The confusion matrices are relatively coloured with the true labels on the vertical axis and the predicted labels on the horizontal axis. The histogram in the bottom of each image uses *ten* bins, whose size is chosen by scipy, relative to the spread of the data. The y-axis or height of each bar of the blue, orange and green histograms represent the relative frequency of examples in the training, test and adversarial classes, respectively. The x-axis represents the score difference between the logit vectors. The parameters described in the captions are large, typical or smaller than typical. These parameters determine the size of the perturbation applied to the image and are provided in full detail in the appendix. These parameter settings are described in relation to the default parameter sets provided by the cleverhans API, which are the typical settings.

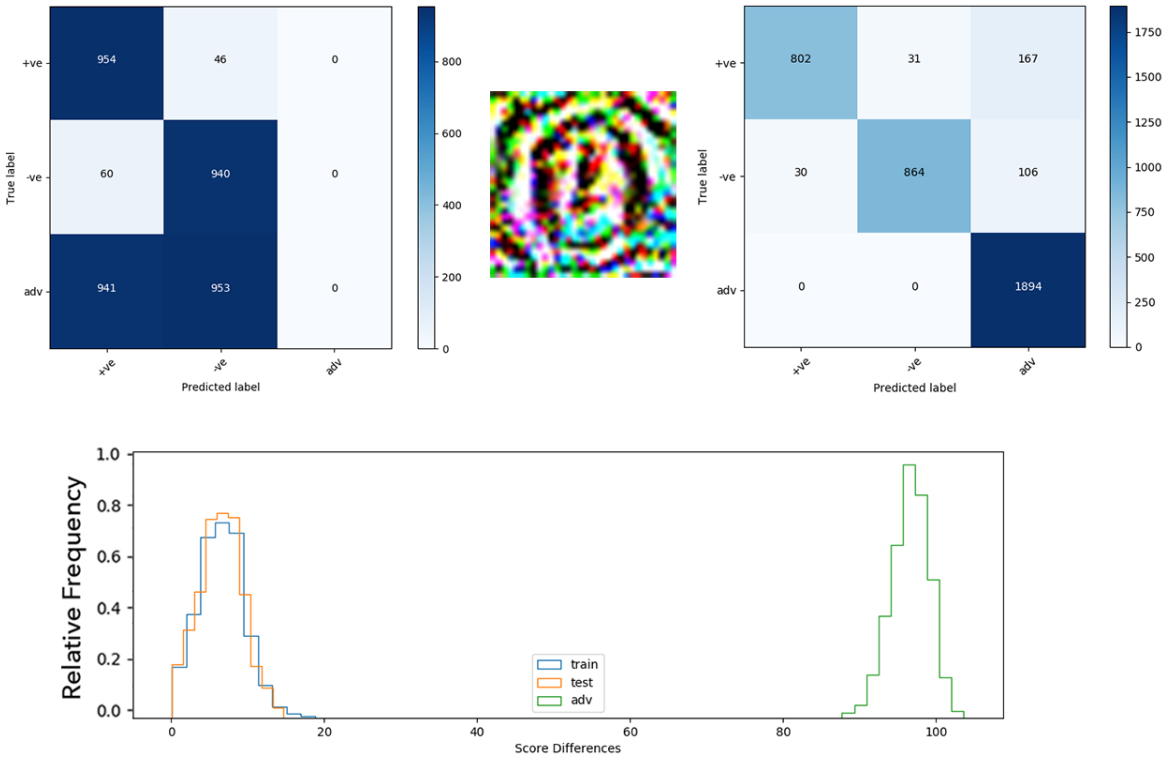


Figure 8: Attack from (Madry et al. 2017) in category 3, otherwise known as the Madry attack, with high parameter settings. The differences between the logits are very high, at least 70, albeit less exaggerated than some other attacks which have score differences in the hundreds. These scores were mainly caused by large perturbation vectors applied to the images. The adversarial images with scores represented by the green histogram are very far away from the training and test data, which means that Background Check will separate them well. The central example adversarial image is completely coloured noise, due to the large perturbation. The left hand confusion matrix, on the bottom row, shows the attack had equivalent ease generating adversarial images from either class. The right hand confusion matrix shows a reduction in average recall due to the many, 198/1000 false negatives, for the first class, and 136/1000 for the second, which, because the classifier is evaluated on the average recall, will make a large difference to the final average recall.

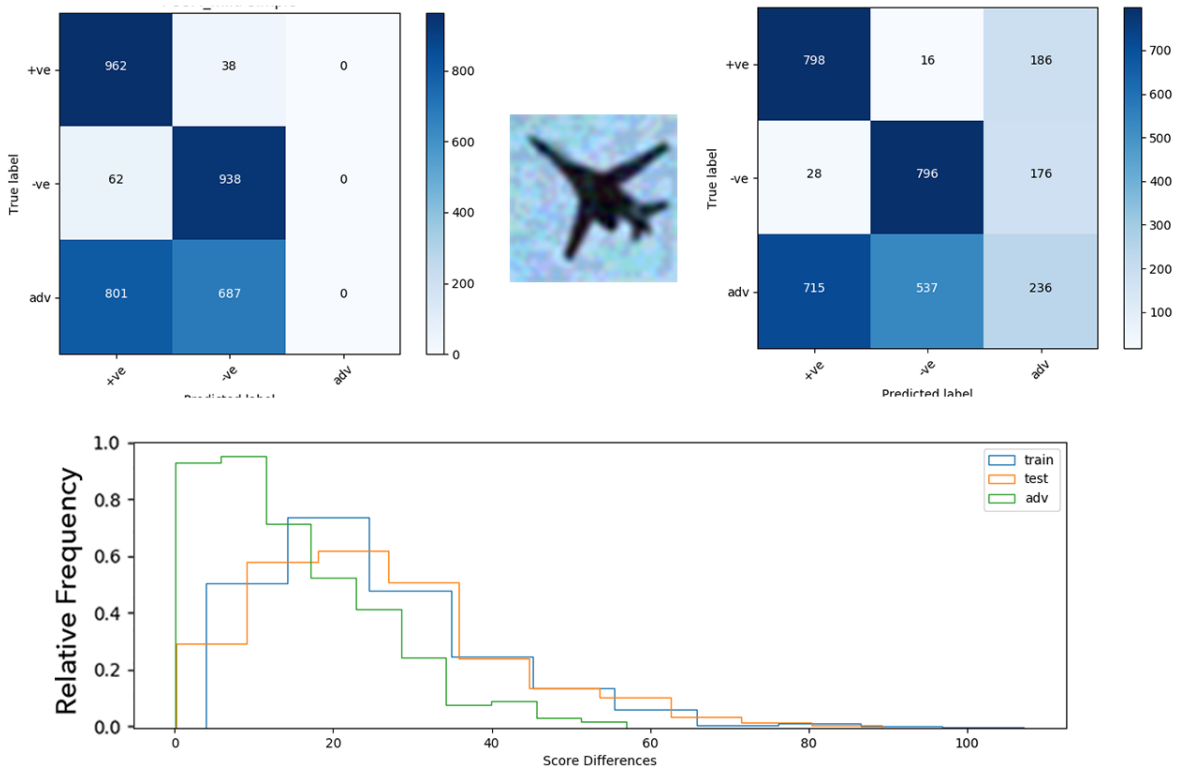


Figure 9: Attack from (I. J. Goodfellow, Shlens, and Szegedy 2014) in category 2, otherwise known as the FGSM attack, with smaller than typical parameters. The analysis is similar to that in Figure 8 except for a few important differences. The first, is that the adversarial examples represented by the green histogram, clearly overlap with the training and test distributions in this space. This makes it very hard for Background Check to separate these examples. This pattern occurred mainly for attacks with smaller than usual parameters. This is exemplified by the right hand confusion matrix classifying only 16% of the adversarials correctly. For more analysis, see the caption in Fig. 8.

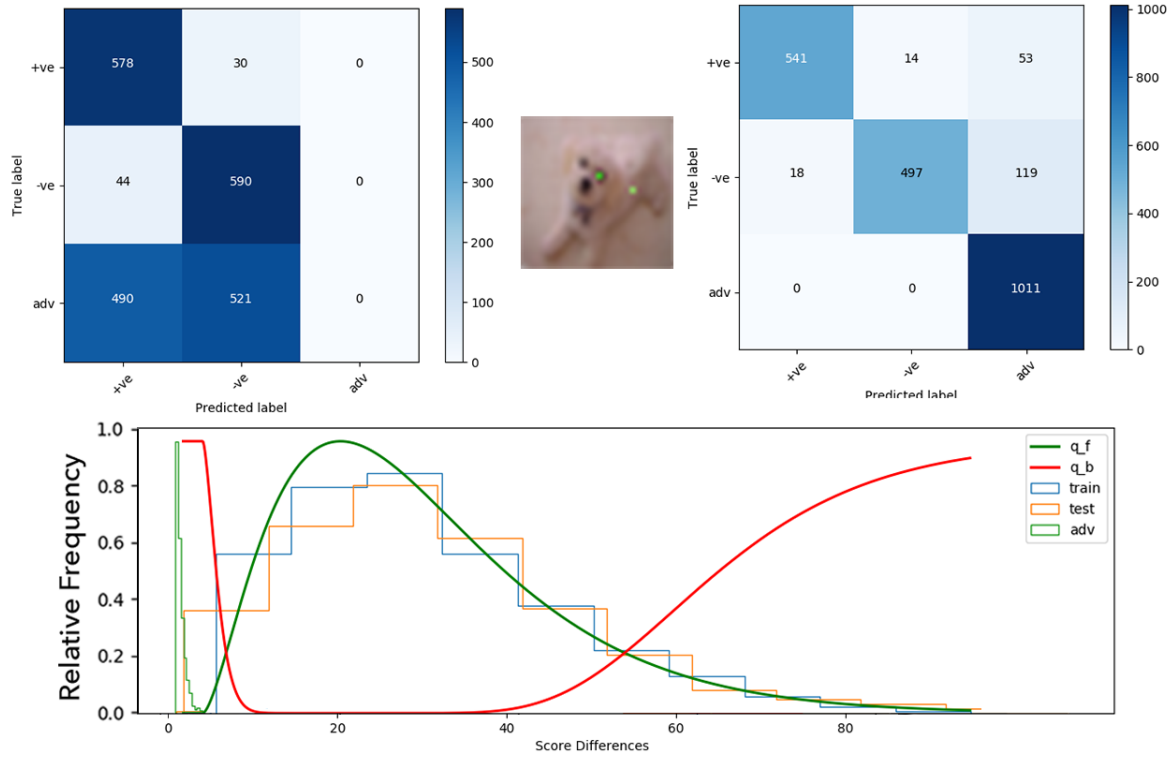


Figure 10: Attack from (Papernot, McDaniel, Jha, et al. 2016), otherwise known as the JSMA attack, in category 1, with typical parameter settings, with q_b and q_f shown. The adversarial differences here are smaller than training and test differences, allowing Background Check to separate them well. JSMA only changes two pixels in this attack.

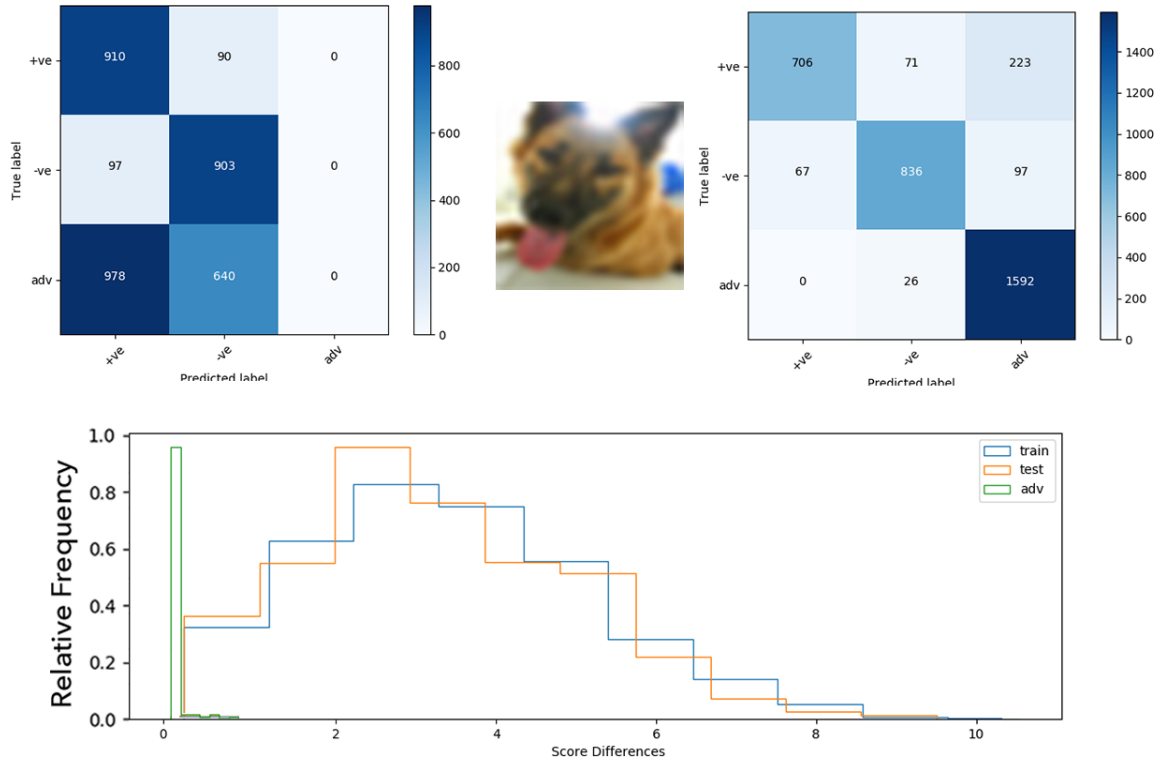


Figure 11: Attack from (Moosavi Dezfooli, Fawzi, and Frossard 2016) (DeepFool) in category 1, with typical parameters. The adversarial differences here are also smaller than the training and test differences, allowing Background Check to separate them well. DeepFool finds the minimum perturbation vector to misclassify an image, hence, it appears the images are close to the softmax decision boundary. For more analysis, see the caption in Fig. 8.

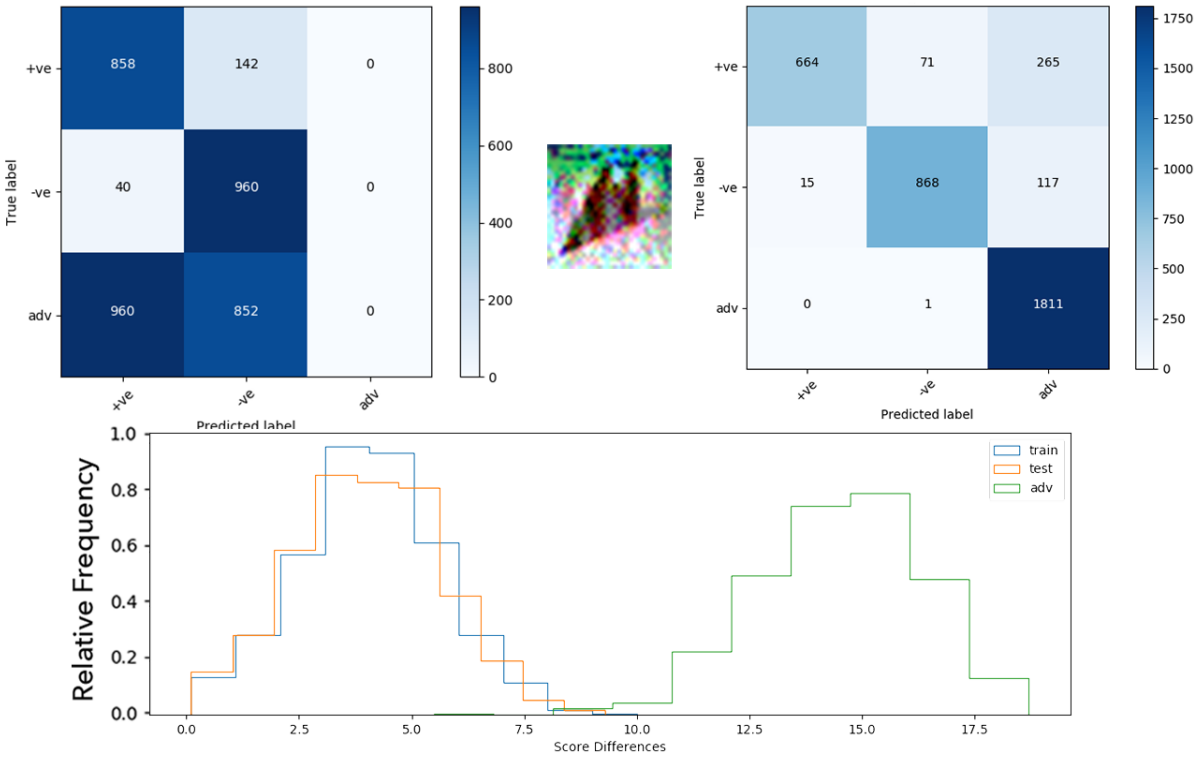


Figure 12: Attack from (Dong et al. 2018), otherwise known as the momentum attack, in category 3, with typical parameter settings. Here, the clusters are separated well, leading to an effective separation by Background Check. For more analysis, see the caption in Fig. 8.

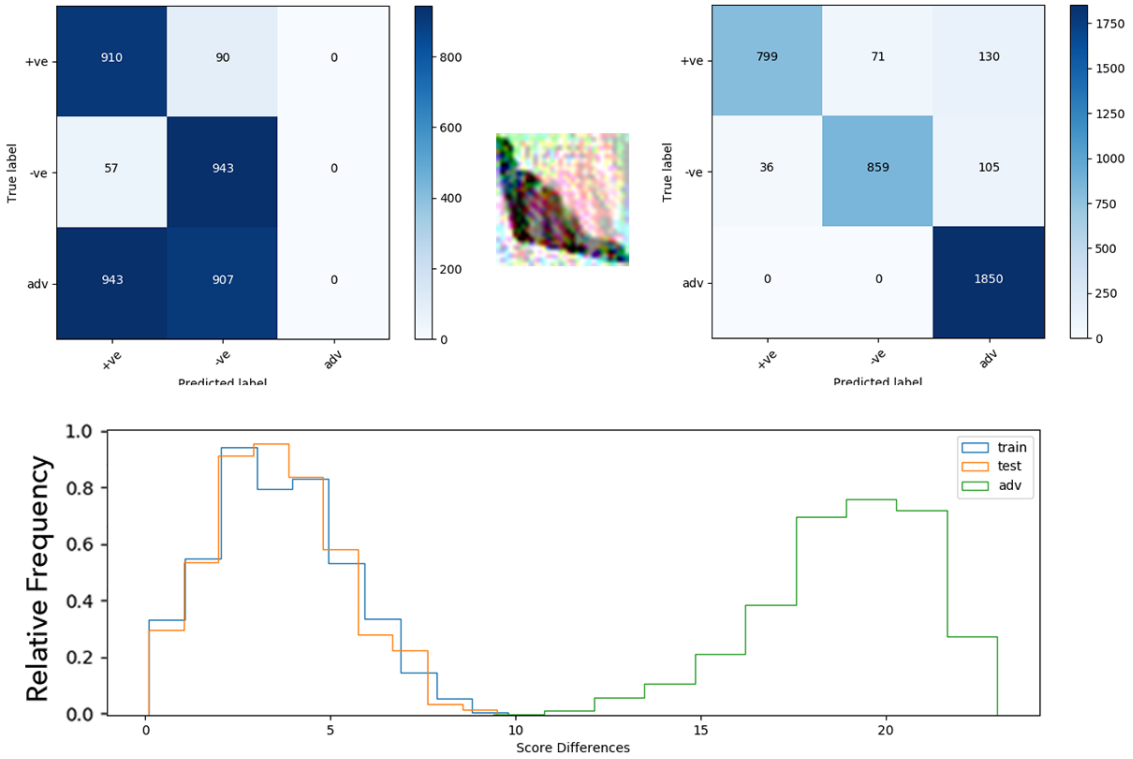


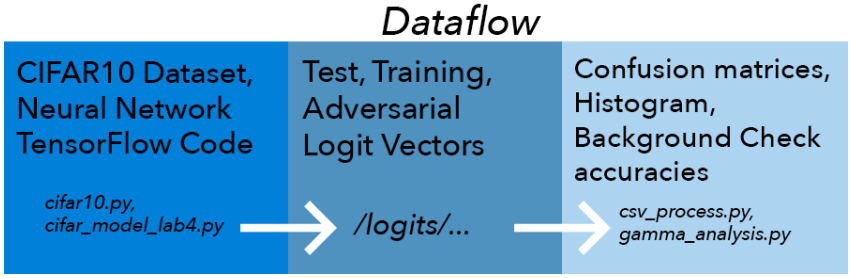
Figure 13: Attack from (Kurakin, I. Goodfellow, and Bengio 2016), otherwise known as the Basic Iterative (BI) attack. in category 3, with typical perturbation. Once again, here, the clusters are separated well, leading to an effective separation by Background Check. For more analysis, see the caption in Fig. 8.

6.3 Table of results

Table of results						
Method	2-class average recall	3-class average recall with Background Check	Difference	Adversarial TPR	Dataset	Regularised
BI Large	91%	81%	-10	100%	DVP	No
BI Typical	93%	88%	-5	100%	DVP	No
BI Small	90%	66%	-24	56%	DVP	No
BI Very Small	91%	58%	-33	21%	DVP	No
DeepFool Typical	91%	84%	-7	98%	DVP	No
JSMA Typical	94%	89%	-5	100%	DVP	No
JSMA Typical	92%	87%	-5	100%	FVS	No
JSMA Typical	88%	77%	-11	84%	AVH	No
JSMA Typical	85%	**87**%	+2	97%	FVS	Yes
Madry Large	95%	89%	-6	100%	DVP	No
Madry Typical	89%	85%	-4	100%	DVP	No
Madry Small	88%	60%	-28	28%	DVP	No
Madry Very Small	91%	53%	-38	23%	DVP	No
VAT Large	94%	71%	-23	47%	DVP	No
VAT Typical	93%	66%	-27	34%	DVP	No
FGSM Large	94%	77%	-17	61%	DVP	No
FGSM Typical	95%	75%	-20	53%	DVP	No
FGSM Typical	91%	88%	-3	100%	FVS	No
FGSM Typical	92%	89%	-3	94%	AVH	No
FGSM Typical	87%	**87**%	0	99%	FVS	Yes
FGSM Small	95%	58%	-37	16%	DVP	No
Mom. Large	94%	89%	-5	100%	DVP	No
Mom. Large	92%	**92**%	0	100%	FVS	No
Mom. Large	87%	85%	-2	100%	AVH	No
Mom. Typical	91%	84%	-7	100%	DVP	No
Mom. Typical	82%	79%	-3	93%	FVS	Yes
Mom. Small	88%	71%	-17	63%	DVP	No
Mom. Very Small	91%	55%	-36	7%	DVP	No
Baseline	50%	33%	-17	33%	DVP	No
No Adversarials	94%	88%	-6	100%	DVP	No

28 Total Attacks	2-class	3-class	Difference	Adversarial TPR
Average	91	78	13	74
Std-Dev	3	12	13	33

Figure 14: This represents the data analysis pipeline that was used to process all of the neural networks. The network was trained, before logits were outputted to files. These logits were then used by Background Check. To see more, please see the GitHub link referenced in the methodology section.



6.4 Pipeline of code and data analysis

The data pipeline can be seen in Figure 14. The data is split into per-class logits, and labelled depending if they are adversarial or not. The majority of the code base is original, neat and relatively easy to read, in addition to being modular. This code can also be found on my GitHub (see Methodology section above). The limited type strength and existence of global side effects in Python makes it difficult to give a rigorous specification for the code base. The average recall, rather than the accuracy was used to evaluate the neural network due to severe class ratio imbalances, which was caused by the differing attack speeds and efficacy. There exist minor caveats surrounding the table of results, which ought to be mentioned. More specifically, in the tables of results, the no adversarials row, corresponds to Background Check applied to data, where the only images used at testing time is exactly the test set. In addition, the table of the 28 attacks excludes the no adversarials and baseline rows. For clarification, the 3-class header applies to 3-class with Background Check applied, in the tables. The appendix also supplies a reference for the names of each of the attacks.

Average recall in the context of this thesis is simply the sum of all of the class specific true positive rates divided by the number of classes. The average recall (AVR) is (34) where the AVR can be found be dividing the true positives by the row marginals, in the confusion matrices, in the figures within the results section, weighted by the number of classes.

$$(\sum_{i}^{C} \frac{TP_i}{TP_i + FN_i}) \times 1/C \tag{36}$$

6.4.1 Materials and Methods

The hardware used to train the neural network model was a Nvidia P100 GPU on the University of Bristol's Blue Crystal Phase 4 supercomputer. The learning algorithm was back-propagation, with a learning rate that decayed every 1000 steps by 80%. The neural network was trained with the Adam optimiser (Kingma and Ba 2014), on the cross entropy loss error metric. The CIFAR-10 dataset was both the training and test set. Initial images

that the adversarial attacks used to create the perturbed images were images from the test set. The network was trained on all ten classes, but tested on only two out of the 10 classes, in the pairs mentioned (DVP etc). Three out of the 28 attacks were performed on regularised networks. Two of these three had L_2 regularisation on the weights applied to them and the third Mom. Typical, had weight dropout with a probability of 0.5 applied. The adversarial attacks all had full access to the neural network, including the trained weight matrix and activation functions. The training and validation ratio is 9:1 and the training, test ratio is 5:1. Each class had exactly 1000 test images and 5000 training images. The number of training iterations was fixed at 300. The images were pre-processed such that all image pixel values were floating point values between zero and one, rather than values between 0 and 255. This pre-processing step was performed because the documentation suggested that the adversarial attacks work more effectively on these values.

6.4.2 Why this particular architecture?

Why was this network constructed and not the latest, highest accuracy, neural network? The main reason is that this thesis aimed to show an existence result in the deep learning literature. Another reason is that there was difficultly in reliably implementing mainstream neural architectures. This was because, I did not find any published TensorFlow neural network models coming from the authors of the papers corresponding to the papers on adversarial defences. This means replicating the models themselves is hard, as model weights are initialised in high dimensional spaces ($\times 10^6$ weights or more) randomly, without a code based random seed. In addition, the papers examined did not mention their full parameter sets, which are often also set randomly, or publish original code accessible to me. (Lipton and Steinhardt 2018) expands on related issues, covering ambiguity in artificial intelligence and machine learning related journals.

6.4.3 Parameter choices: Large, Typical, Small, Very Small

Cleverhans provided a set of parameters that were ambiguous, partly due to the complexity of the code and partly due to the differing attack implementations. As such, the parameters were chosen such that the resultant images from the application of the attack fell into one of the following four classes:

1. *Large* - Image consisting entirely of colourful noise.
2. *Typical* - Moderate noise levels, underlying image recognisable.
3. *Small* - Recognisable noise, but somewhat clear image.
4. *Very Small* - No noticeable noise, crystal clear image.

Actually getting the attacks to create these images in the precise classes was a little haphazard, but on the whole, was achievable. The language used in this thesis uses large perturbations and large parameter choices interchangeably. Due to time constraints, I was

unable to iterate through all four classes for each attack type. For instance, on Blue Crystal Phase 4, JSMA took around 30 seconds per attack, which is infeasible for many thousands of images. The parameters, for each attack type, are described in the appendix, along with the papers corresponding to each of the attack names, given in the results table.

6.5 Separating images based on the relative difference of logits

Another idea that I had during the project, was to separate the adversarial images from the test and training images using the ratio between the output logits in the two-class case. Figure 16 shows this precise ratio of neuron values for over 1,500 images. It is clear, that given more time, this could be an additional dimension to separate adversarial data from test set data. Adversarial data seems to find very rigid ratio values, whereas the test data takes on a flatter, wider shape across many different ratios.

6.6 Discussion of results

The results show a strong performance of Background Check, in rooting out adversarial examples, even if the average recall reduces for almost all of the attacks. This notion is normalised when considering that all of the adversarial defences I saw in the literature, had a consistent reduction of accuracy. Neural networks with no ability to classify adversarial attacks, can achieve at most 66% average recall. If measured with accuracy, this figure would reduce to zero by simply increasing the class proportion of the adversarial images. On average, average recall reduces by 13%. Out of the 28 attacks, 3 of them managed to achieve an average recall greater than or equal to the original average recall, which so far, I have not managed to find in the literature on adversarial defences. The attack types which saw the largest reductions in average recall, were the attacks with small perturbation vectors applied to them. This is exemplified by reductions in average recall from the 2-class to the 3-class setting of -37, -38, -33 and -36 for three very small perturbations and one small perturbation. This is approximately 2 standard deviations from the mean of the differences, in quite a skewed data set, which has a mean of -13. The baseline measure was created by choosing classes with uniform randomness. The 2-class results consists of only the positive and negative classes, excluding all predicted and true values for the adversarial class.

On the other hand, as the attack parameters increased and the resulting perturbations increased in size, the per-class logits distanced themselves further from each other and the images decomposed to coloured noise, with no visible CIFAR-10 object class. These large parameter settings tended to place the adversarial images in category 3, which allowed Background Check to separate them well from the test and training data. This occurred most notably for the BI, Madry and Momentum attacks. In Figures 18, 19 & 20, this trend can clearly be seen. An educated guess from a process point of view, is that the gradient based algorithms underlying these attacks, only update when the confidence for the target class increases. Figures 18 to 20 detail the relationships between logit differences and parameters for

Figure 15: This picture encapsulates many core ideas in this thesis. As the difference of two output neurons tends to infinity, softmax and temperature scaling both tend to 1. Empirically, this paper finds that colourful nonsensical noise exists in regions corresponding to large pairwise differences. Adversarial attacks show that class instance spaces include vast regions of nonsense, revealing the fallibility of the softmax function in classification with limited training examples for this network. Background Check acts as a stronger assumption over the space by only quantifying knowledge over the test and training regions which empirically tend to overlap. In the image $r(x)$ is the red line (q_f) divided by the orange line (q_b) plus insignificant ϵ to ensure safe division. The green regions represent the confidence in the two class case, that temperature scaling or softmax will give with the respective score differences at the bottom of the page.

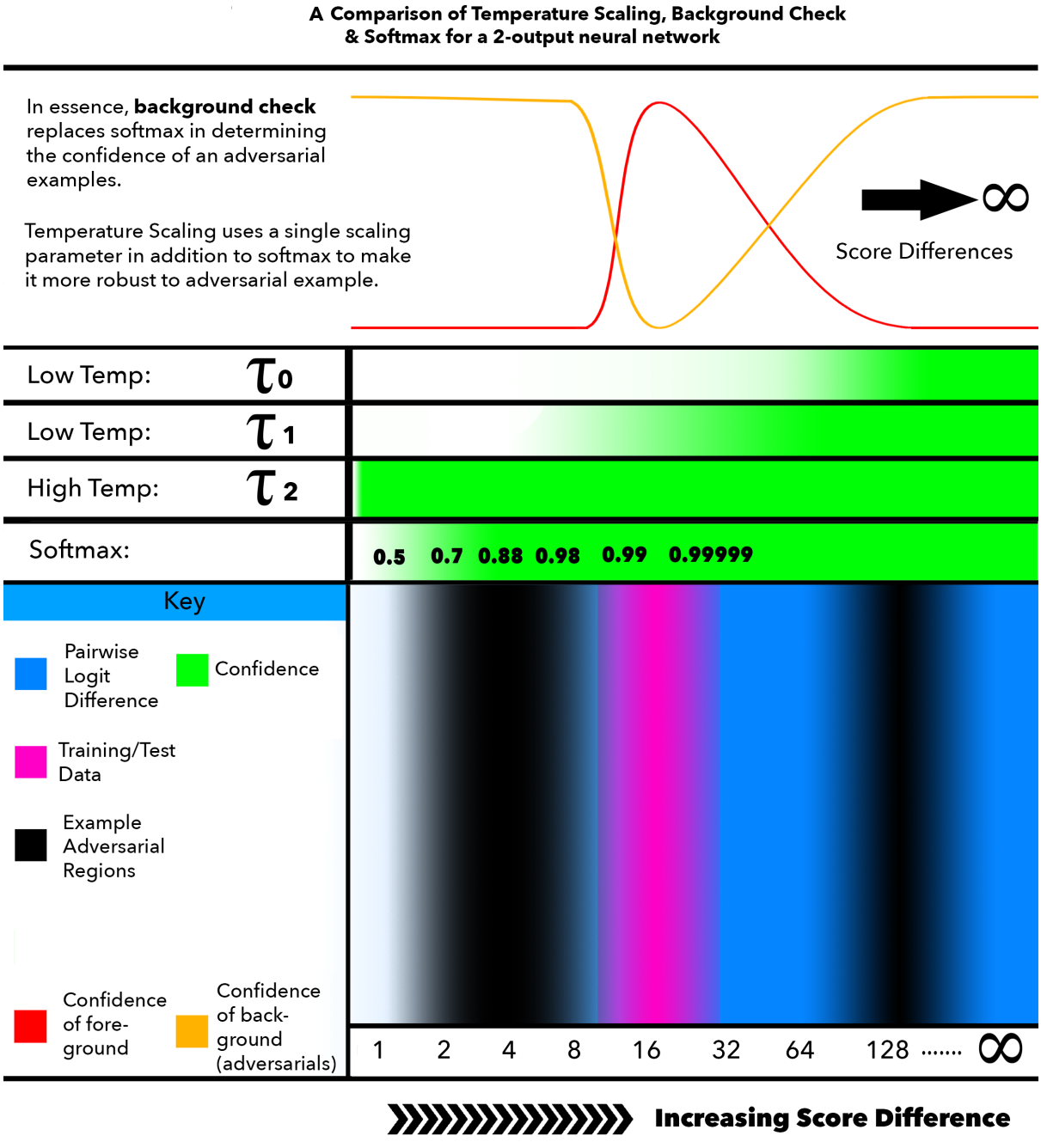
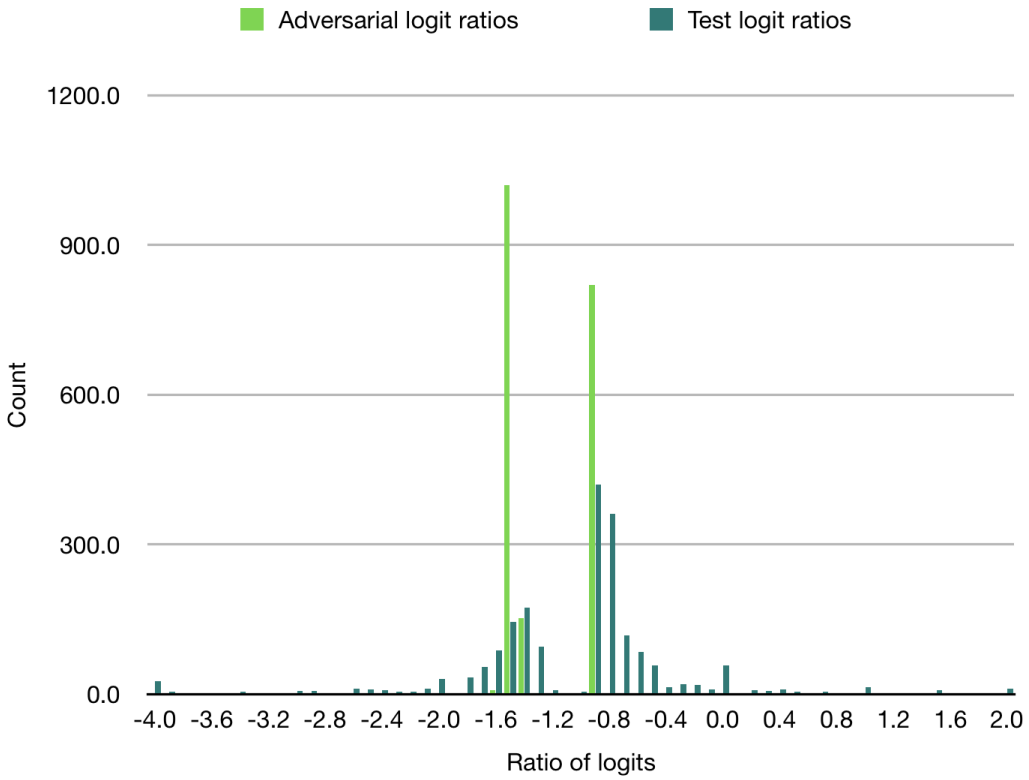


Figure 16: This figure demonstrates the ratio of the adversarial logit vectors for the Madry attack with typical settings, as well as the test set logit ratios, for over 1500 images. There are clusters around -1 and 1 because these correspond to one direction i.e. dog versus plane and the opposite direction i.e. plane versus dog. There are clearly differences in the distributions, in that the test ratios have a flatter distribution. The adversarial ratios seem to only hold to values of -0.9, -1.5 and -1.4 exclusively. The test ratios go so far as having ratios of -4 or -2.



these particular attack types. The corresponding difference between the 2-class and 3-class average recalls is only -10, -6 and -5, for the three attacks, which is much smaller than the mean of -13. Generally, the best results occurred for those parameter settings that were in the range of either large or typical. Two out of the three results that had the best results, i.e. differences of zero or above, were also attacks performed on networks that had regularisation techniques applied to them.

For some attacks, large parameter combinations did not make it easier to identify adversarial examples. Two examples demonstrating this, are firstly the FGSM Large attack, whereby increasing the parameters saw no increase in average recall relative to Typical parameters, and secondly the BI Large attack saw a decrease of 7% relative to typical parameters. An overview of the relationship between parameters and accuracy is given in Figure 17. As the variation is high on the same parameters on different datasets, such as in JSMA, it is difficult to rule out this reduction for the BI Large attack, as an effect of random variation stemming from the random weight initialisation. Please see the appendix of additional figures to see all attack distributions not presented in the results section.

The no adversarial row has a reduction in average recall, even assuming an adversarial TPR of 100% because the introduction of Background Check creates false negatives at the edge of the distributions where q_b and q_f cross over, in the training and test distributions. Performing an experiment on a network with this characteristic, revealed a reduction of average recall of 6% on a network which had no adversarial examples.

6.7 Evaluation of Background Check in the context of adversarial defences

6.7.1 Disadvantage of methodology and caveat of results

Given more time, the Background Check based method should be tested against contemporary reactive defence methods. As very few papers use 2-class average recall, it is very difficult to put this method in context of the wider literature on adversarial defence strategies. It is sometimes easier to achieve high average recall on fewer classes in the context of classification, because as the scope of the problem increases, the ease with which a high true positive rate can be achieved in each class, often, reduces. It would be worthwhile to scale up the number of classes this method is tested on at any one time.

This method does not work for more than two classes, as the dimensionality of the logits increases and the L_1 measure discussed makes no sense in this space. This method was not imbued with a way to work in higher dimensions, as it was unclear at the beginning of this thesis, what a good measure would be to replace the logit differences in higher dimensional spaces. Usage of the maximum mean discrepancy or the energy distance could have been a worthwhile experiment.

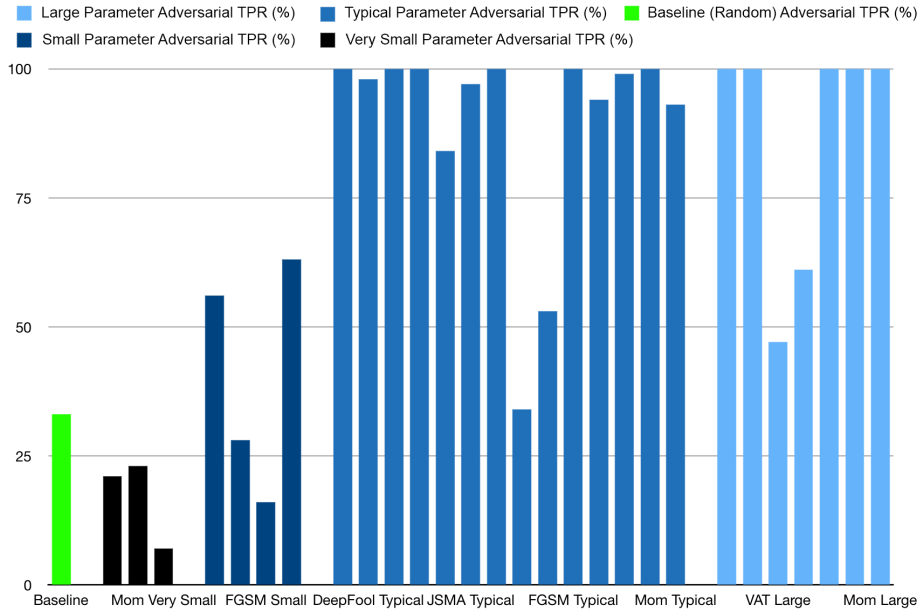


Figure 17: As the perturbation vector increases resulting in noisier images, Background Check separates the adversarials from the test and training sets more effectively, resulting in an increasing adversarial true positive rate. The worse performance occurred when smaller perturbations were used and the adversarial logit difference distributions started to merge with the test distributions as in the black examples, which performed even worse than the random baseline.

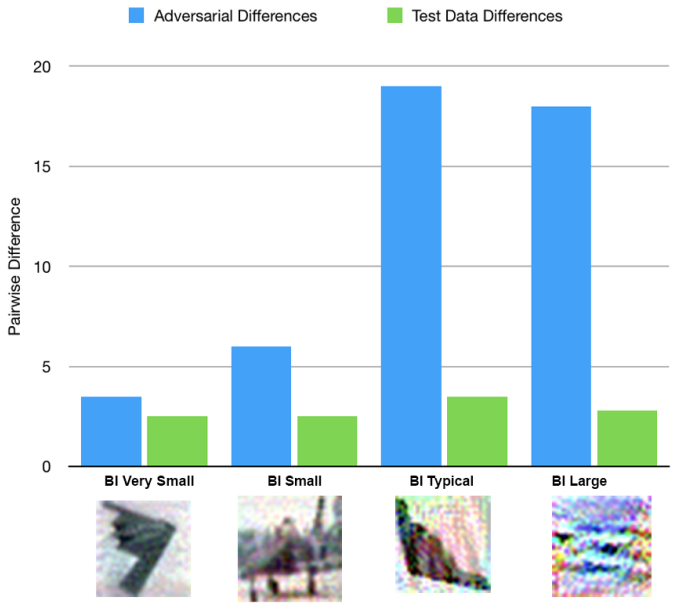


Figure 18: The pairwise difference scores for the basic iterative method. This demonstrates that the differences increase massively as the perturbation vector applied to the image becomes larger. The test data differences are shown next to the adversarial differences to act as a baseline.

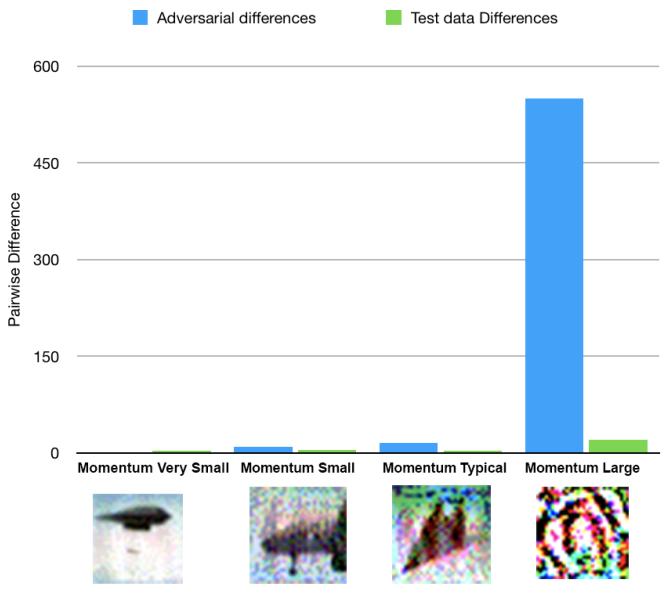


Figure 19: The pairwise difference scores for the momentum method. This demonstrates that the differences increase massively as the perturbation vector applied to the image becomes larger. The test data differences are shown next to the adversarial differences to act as a baseline.

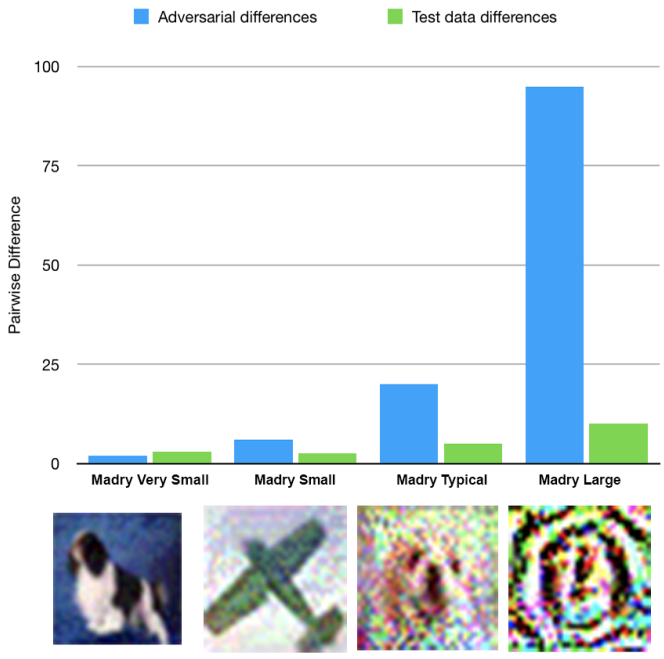


Figure 20: The pairwise difference scores for the madry method. This demonstrates that the differences increase massively as the perturbation vector applied to the image becomes larger. The test data differences are shown next to the adversarial differences to act as a baseline.

Ultimately, Background Check is a calibrating procedure, yet in the context of this thesis the main usage was using the reliability factor to determine classifications for the background or foreground classes. In future, it would be worthwhile to look at the calibration performance, using calibration and refinement loss to understand the quality of the probability estimates that the neural network outputs with Background Check.

The approach taken does not necessarily utilise Background Check to its fullest extent. One could imagine a scenario in which each layer of a neural network has multi-dimensional Background Check applied to it. Background Check would look for outliers after the activation functions of each layer. However, an argument against this methodology is that it could restrict the generalisation capacity of the neural network, by restricting each layer to only exhibit distributions seen before in the training data.

This thesis has not developed a fully grounded theoretical understanding of the relationship between logit vectors and adversarial examples and as such further work is needed there. This means that the empirical result achieved in this paper does not necessarily generalise to other attack parameters. Additionally, there is no theoretical work justifying whether outliers on the score distribution are necessarily adversarial.

The method proposed only works if the outliers are assumed to be adversarial examples, which may be a haphazard assumption. This approach is only adversarial detecting but not adversarial correcting. In practice, in safety critical applications, if an example was in the outlier class it could be labelled as cautious and a human could manually classify those images.

One could envisage a better way of measuring the success of Background Check. (Condessa, Bioucas-Dias, and Kovačević 2017) proposes performance measures for classification systems with the rejection option. These measures consist of metrics such as the *nonrejected accuracy*, which measures the ability of the classifier to accurately classify nonrejected samples. The *classification quality*, which measures the correct decision making of the classifier with the rejector and finally, *the rejection quality*, which measures the ability to concentrate all misclassified samples onto the set of rejected samples.

Finally, the way in which parameters were chosen in future would have been more systematic. The appendix shows that the parameter settings were rather arbitrary and ultimately up to my judgement as to the class that the images fell in. This is not in line with the principles of the scientific method as there is scope for choosing parameter values that match what one wants to see to obtain a result. This did not happen in this case, but should be ruled out if this were to be repeated.

6.7.2 Advantage of methodology

An adversarial defence strategy has been created that is both reactive and can detect, but not correct, adversarial examples. The most interesting addition to the literature on adversarial defences, is that in the majority of defences against adversarial attacks, as the size of the perturbation vector increases, the defences find it harder to defend against the examples, a relationship detailed in 6, yet in this defence mechanism, the network finds it easier to defend against higher attack parameters as they exhibit more extreme values in the final output layer. In addition, other methods will likely attempt to recover the original object class of these data points, which is a mistaken strategy, as the data points themselves appear to be nonsensical noise. I argue images which have large perturbation vectors should not be ignored as much as they are in mainstream literature, as these images can occur and be just as damaging in the real world.

Additionally, in the literature mentioned in the section on adversarial defences, many defence methods use adversarial examples in the training procedure. This requires knowledge of the adversarial method at training time, which is an impractical assumption. In addition, many different attack parameters must be used in training for these methods. This initial preparedness is in some sense an unfair advantage against the method I propose and thus imbues those methods that require training with a rigidity that this method does not have.

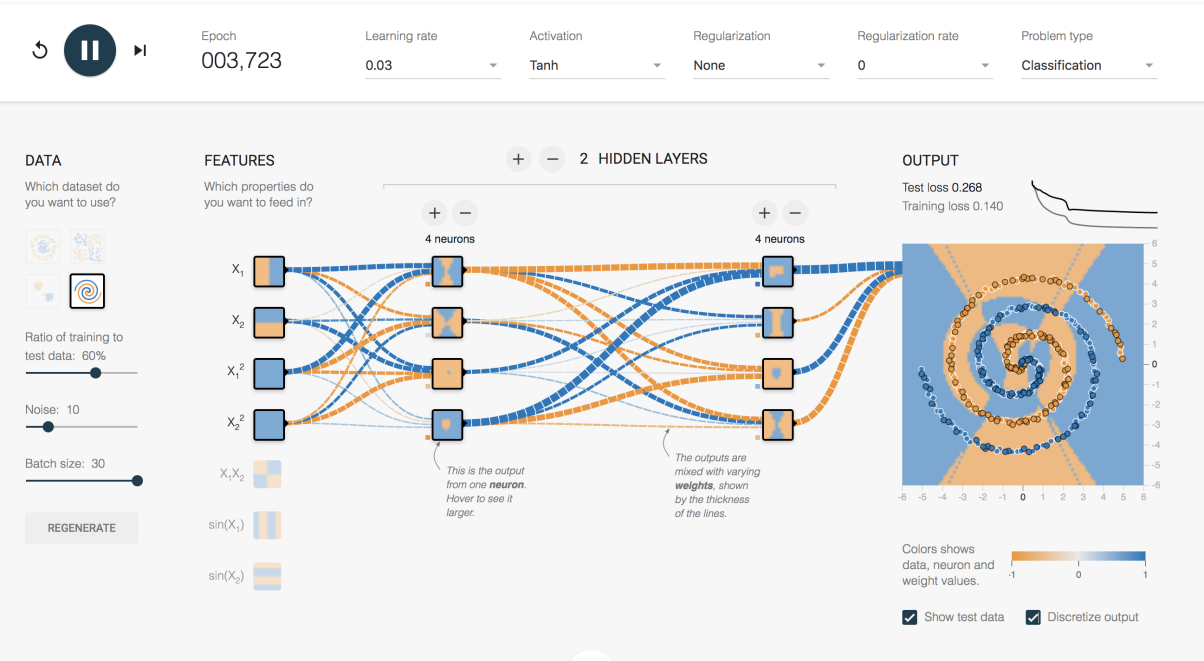


Figure 21: A neural network whose structure entails poor classification boundaries. Source: playground.tensorflow.org

7 Analysis: Limitations of neural networks

In order to better understand adversarial defences and to understand the context of future work, a deeper analysis is needed. In order to explain the limits of Background Check, naturally the limitations of neural network methodologies will be discussed. The limitations of neural networks have been recognised in a wide variety of contexts (Gee and Prager 1995), (Malthouse 1998), especially when dealing with highly structured, often compositional, yet limited data sets (see "In defense of skepticism about deep learning" - Gary Marcus). However, from the work on adversarial attacks, it is clear that there are fundamental issues with the probability estimates coming from neural networks. There needs to be a distinction between probabilities for data close to the decision boundary, versus data very far away from any training data seen. This imbues models with a better way to know when they don't know. This is important when dealing with data, where object classes that don't even have a representative output neuron, appear in the test set.

Whilst the universal approximation theorem (Hornik, Stinchcombe, and White 1989) proves that neural networks with an arbitrarily large hidden layer coupled with a softmax output layer can approximate any function arbitrarily well, it is not necessary that a high accuracy on the classes seen entails low probability estimates on true negatives, a fact empirically justified in this thesis by the existence of adversarial examples on networks with a high average true positive rate.

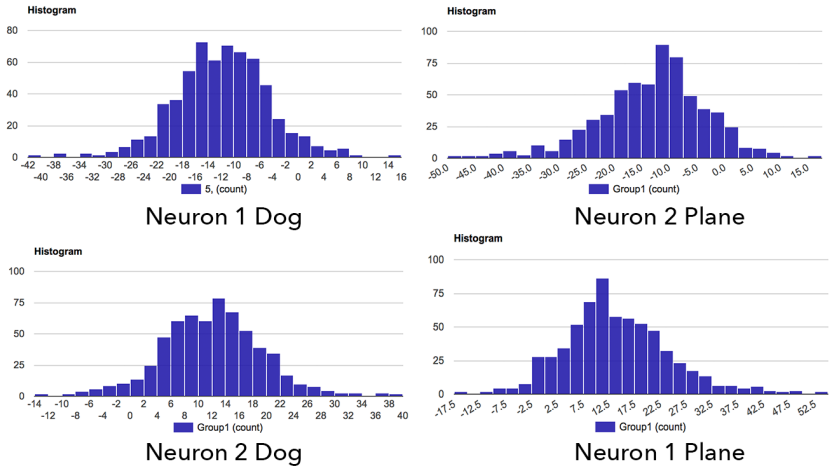


Figure 22: This shows the distributions for the logits.

Adversarial examples can even be caused by poor designs of neural networks, in particular, overly simple network architectures, leading to poor classification boundaries, seen in Figure 21. There clearly exist numerous points classified in the wrong class, in this cherry picked example from playground.tensorflow.org. The adversarial examples could therefore be a product of a lack of expressive power of the architecture. A theoretical testament to this is (Neal 1996), who proves that an infinite number of hidden units, for many neural networks with fixed hyperparameters, are required for neural network models to converge to a Gaussian process prior over functions.

7.1 Does high accuracy mean that the logits are Gaussian?

In order for softmax to calibrate the logit vectors effectively, the one-dimensional per-class classifier scores must be Gaussian. Evaluating the Shapiro-Wilks test, a frequentist test of normality on over six-hundred data points per class, reveals that even with high average recall, the data is not necessarily normally distributed.

H_0 is the data is normally distributed.
H_1 is the data is distributed from another distribution.

The following table lists the p-value percentages associated with the chance of rejecting correct H_0. The alpha value is 5%.

Neuron	Dog	Plane
Neuron1	0.50%	0.0014%
Neuron2	2.27%	0.0014%

The difference between the data samples and the normal distribution are big enough to be statistically significant. As the logits are not normally distributed, softmax will not calibrate these logits correctly.

7.2 Commentary: Why are accurate probabilities fundamentally hard to guarantee?

Adversarial inputs are often explained in terms of linearities or non-linearities of the network. In this section, a theoretically grounded quantification of model types will be given with respect to adversarial robustness. In order to provide this quantification, probability estimates must be dissected. Probabilities for the purposes of this commentary are defined (see (Kendall and Gal 2017)) with respect to the first two following notions, with the third ignored for the sake of this quantification.

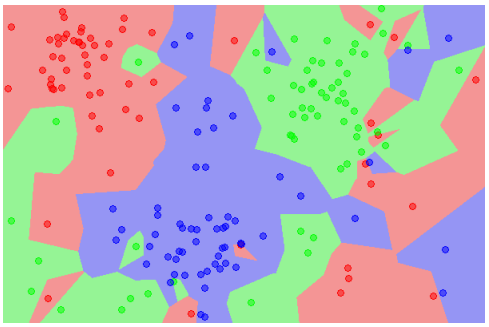


Figure 23: Some papers argue that the non-linearities in a neural network produce complicated decision boundaries where adversarial examples lie.

1. *model uncertainty* = the inherent uncertainty in using the model at hand.
2. *epistemic uncertainty* = the uncertainty of the model parameters.
3. *aleatoric uncertainty* = the noise inherent in the observations, which is decomposed into *homoscedastic* noise and *heteroscedastic* noise, where the former is noise which stays constant amongst inputs and the latter is noise which changes depending on the input type.

In order to calculate the posterior probability of a data point the following must be calculated.

$$P(\theta \mid D) = \frac{P(D \mid \theta)\, P(\theta)}{P(D)} \tag{37}$$

where P(D) is the marginal likelihood (evidence).

The difficulty in computing the conditional distribution arises when computing the marginal likelihood. The marginal likelihood has the form of equation 38 in the discrete case.

$$P(D) = \sum_{i}^{N} P(D_i \mid \theta)\, P(\theta) \tag{38}$$

The difficulty arises as the size of N becomes very high, in computing the posterior (37). This analysis even assumes, that the likelihood $P(D_i \mid \theta)$ for each data point can be calculated quickly. For epistemic uncertainty to tend to zero, (38) must additionally iterate over all

parameters of the model and for the model uncertainty to tend to zero, the sum must iterate over all possible neural networks, with a prior over models and parameters, which is sufficient, due to the universal approximation theorem. Therefore, to expect neural networks to be completely foolproof with such a small dataset, relative to N, may be naive.

7.3 Generative versus discriminative approaches

Neural networks are mostly treated as discriminative classifiers, though there are some exceptions. These discriminative approaches approximate the conditional distribution by minimizing error on a proxy function such as the cross entropy loss (Richard and Lippmann 1991), without explicitly modelling the likelihood function or the marginal likelihood. Neural networks have a temperature scaled softmax function to model the decision boundary. Generative models model the entire joint distribution. Even though generative models capture far more data, (Ng and Jordan 2002) empirically finds that discriminative models are still better at classification.

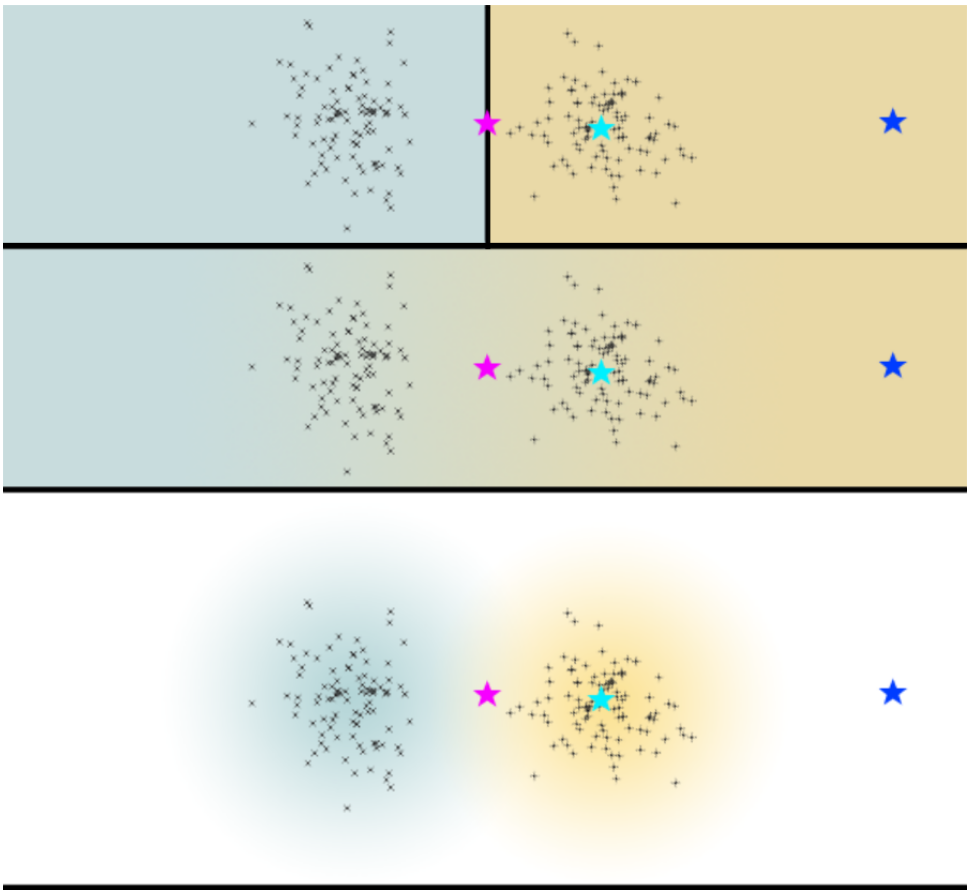


Figure 24: This figure portrays the three types of model being analysed, on two Gaussianly distributed point clusters. The top depiction above the first black dividing line, is a discriminative model, the second is a discriminative model coupled with a softmax-style decision boundary model and the third is a generative model under a Gaussian assumption. The three points; purple, turquoise and blue are used in analysis.

Figure 24 demonstrates the difference of three crucial model types on Gaussian clusters. In order to better understand adversarial examples, this figure is helpful. In the case of the discriminative model, adversarial examples can be found if a reliable method exists for finding examples from the light-blue class in the region pointed out by the blue star. Examples from regions around the turquoise star will be well classified and examples by the purple star will be poorly classified as high confidence will be allocated to a uncertain region equidistant from both clusters. The decision boundary method optimises for points close to the purple point by providing uncertainty associated with predictions. The decision boundary method can still suffer, if the decision boundary method is not flexible enough. The generative model is the most useful out of all three as the model knows what it doesn't know, if one assumes a value of zero means don't know rather than a rejection. This generative model is superior, because, as well as correctly predicting the turquoise and purple points, the blue point will receive a low probability for both classes.

In higher dimensions, weaknesses of all model types with respect to adversarial attacks arise. Problems with the discriminative model arise when vast regions of space in which no training data has been observed are classified with no understanding of the uncertainty involved, as a particular class. The size of these spaces grows exponentially with the number of dimensions, due to the curse of dimensionality. For the decision boundary method, for each dimension, the assumption underlying the distributions at the decision boundaries may not hold for each class. An example of this would be if a particular class had all of its data points in a pattern similar to the typical distortions produced by naive bayes (probabilities pushed towards 0 and 1), in Figure 24, which would clearly undermine the use of a softmax decision boundary. Another example would be if the variances of each Gaussian cluster were massively different, then a single gradient for the classes would be inappropriate such as is potentially the case with temperature scaling. The generative model will fail in higher dimensional spaces if the assumption underlying the data generation process is wrong, meaning large spaces, will potentially include the potential for adversarial example generation in many if not all of the classes.

Analysis of the higher dimensional decision boundary methods, motivates the introduction of an extension to temperature scaling. This minimal extension to temperature scaling, would have $|C|$ scaling parameters, one for each output class $C_i \in \mathcal{C}$ optimized for in a one-versus-rest manner. One could also envision a parameter for each pair of classes, but this would scale with with the square of the number of classes which is impractical given the growing number of outputs in recent neural networks. The class specific scaling parameters can be applied in precisely the same way as τ in temperature scaling. Due to the enhanced expressive power, this may have better calibration performance than temperature scaling alone.

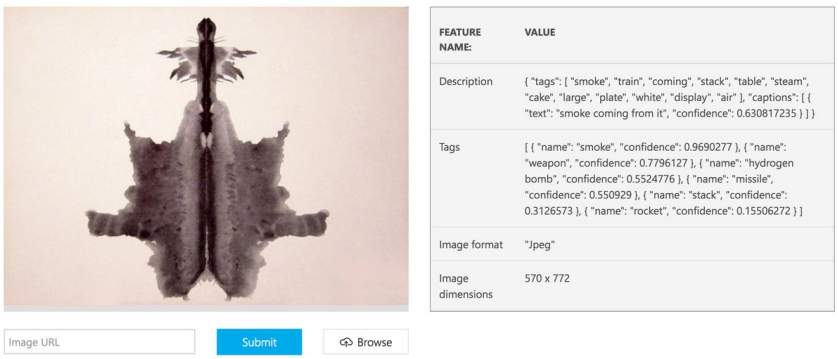


Figure 25: Weapon? Hydrogen bomb? If the CWA is enforced, white noise or Rorschach tests can easily be mis-classified. Source: Twitter.

7.3.1 Why is it important to know what you don't know?

Vapnik advised that for many problems, one only requires the solution of a simpler problem i.e. finding the classification decision boundary rather than generating the full joint distribution. However, more complex problems, may need more consideration. The case for knowing what you don't know is strong when one considers that one also does not necessarily have a model for what you do know. This is often the case in computer vision where images of an object can look completely different when the objects themselves are rotated etc. Ultimately adversarial attacks are the main reason to be wary of the issue of poor understanding of high dimensional spaces. Figure 25 shows that even the Microsoft Azure AI platform is fallible with ambiguous images. If Microsoft contracted for the public sector, images like these could start a war if one didn't have rigorous checks and balances, which in some sense defeats the point of AI in safety critical areas. The utility of neural networks clearly depend upon the specific costs of admitting an adversarial attack.

7.4 Impact - i-LIDS - An example of adversarial risk in the public sector

Figure 25 demonstrates the very real risk of adversarial examples in the context of high risk scenarios such as intruder detection for critical infrastructure. The UK Government sponsors intelligent detection methods for key infrastructural points e.g. i-LIDS[8]. These detection systems likely use neural networks that may be vulnerable to contemporary adversarial black-box attacks and as such their safety can be compromised. Figures 27 to 29 demonstrate the level of detail required by the detection systems.

[8]The Image Library For Intelligent Detection Systems

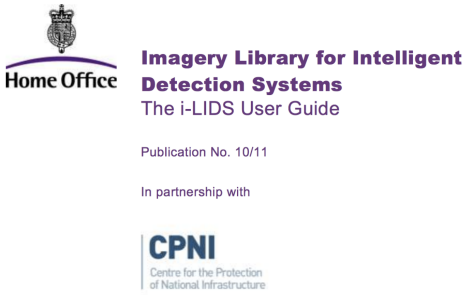


Figure 26: The Home Office accounts for the UK defence and CPNI is involved in protecting key resources such as water integrity through to nuclear security.

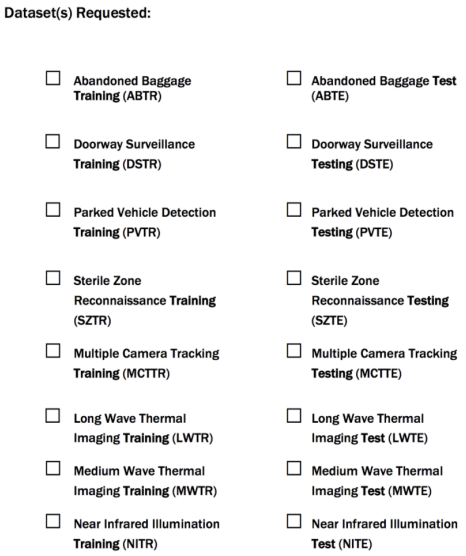


Figure 27: The types of data the government is aiming to use with intelligent detection.

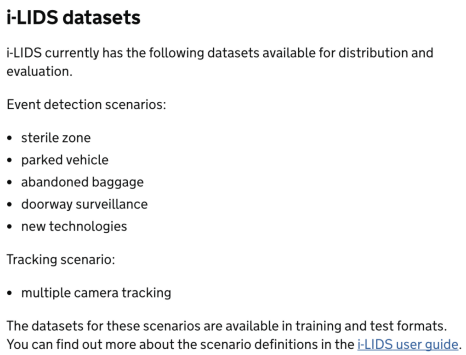


Figure 28: These are scenario types that need to be detected.

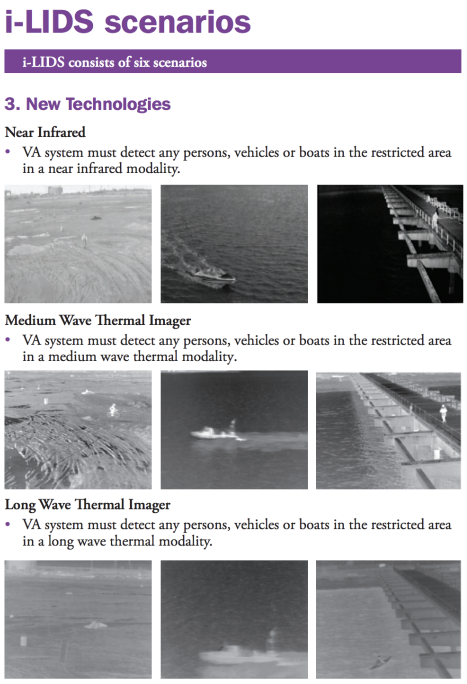


Figure 29: This shows explicitly the richness of detail in classification that is needed.

8 Future work

Relevant future work consists of understanding theoretical reasons for the existence of adversarial examples in the outskirts of the logit difference distributions. Research efforts could also be focussed on the adversarial examples that exist within the logit difference distributions corresponding to the training and test set data. This could manifest in the use of generative models, rather than just discriminative models. Adversarial correction, not just adversarial detection, could also be researched, with options to reject images should they be too noisy. In addition, higher dimensional metrics suitable for high dimensional output vectors could be investigated using the energy distance or mean maximum discrepancy. The extension to temperature scaling could also be tested in the context of calibration. The calibrating performance of Background Check, in the form proposed could be measured, to better assess the probabilities from neural networks. There are other attacks on neural networks that produce images such as those seen in Figure 30, (Nguyen, Yosinski, and Clune 2015). Finally, the results showed that two of the three networks that actually improved accuracy by zero or more in the 3-class case were regularised networks. As such, it would be useful to explore the effect of regularisation on the logit vectors of neural networks and to then test if the approach based on Background Check is more effective on regularised networks.

9 Conclusion

A novel approach to defending neural networks against adversarial attacks has been established, like no approach seen in the literature. This approach intersects two previously

unrelated fields of machine learning, calibration and adversarial defences, using principles seen in the literature on calibration, in particular, Background Check. This thesis demonstrates that adversarial attacks, produced as a result of large perturbations of various forms, can be detected and assigned to an adversarial class. The larger the perturbation, the easier it was for the attacks to be detected, a result previously unseen in the literature. The optimality of softmax in calibration, on neural networks, has raised new doubts, in the analysis, where the Shapiro-Wilks test, demonstrates the logits are not Gaussian, even on high average recall networks. An extensive suite of tests has been performed on various attack types in very recent literature, and the method achieves a strong adversarial TPR of 74% and an average accuracy of 78%, $\sigma = 13$ in the 3-class case and 91%, $\sigma = 3$ in the 2-class case. Whilst, the generalisation of this result is not clear, especially with respect to other network types, the future work section paints a colourful series of next theoretical and empirical steps.

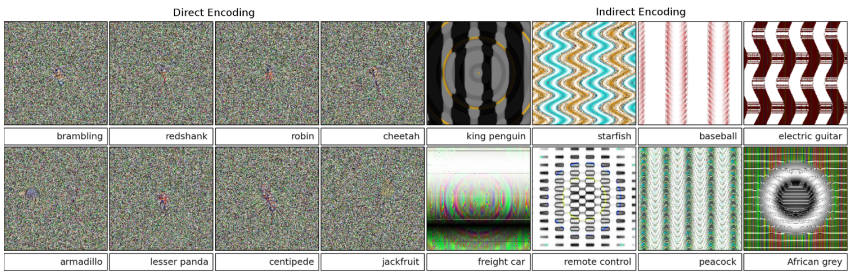


Figure 30: Adversarial images produced by an evolutionary approach (Nguyen, Yosinski, and Clune 2015).

Appendices

A Names of attacks

Attack Type	Paper
BI (Basic Iterative)	(Kurakin, I. Goodfellow, and Bengio 2016)
DeepFool	(Moosavi Dezfooli, Fawzi, and Frossard 2016)
Madry	(Madry et al. 2017)
VAT	(Miyato et al. 2015)
FGSM (Fast Gradient Sign Method)	(I. J. Goodfellow, Shlens, and Szegedy 2014)
Momentum	(Dong et al. 2018)

B Parameters

Attack Type	**Parameter Combination**
BI Large	(nb : 10, ϵ : 0.8, ϵ_i 0.05)
BI Typical	(nb : 10, ϵ : 0.3, ϵ_i 0.05)
BI Mild	(nb : 10, ϵ : 0.1, ϵ_i 0.05)
BI V Mild	(nb : 10, ϵ : 0.06, ϵ_i 0.05)
DeepFool, JSMA	Default
Madry Large	(nb : 150, ϵ : 8.0, ϵ_i 0.03)
Madry Typical	(nb : 40, ϵ : 0.3, ϵ_i 0.01)
Madry Mild	(nb : 40, ϵ : 0.1, ϵ_i 0.01)
Madry V Mild	(nb : 10, ϵ : 0.05, ϵ_i 0.01)
VAT Large	ϵ : 12.0
VAT Typical	ϵ : 1.0
FGSM Large	ϵ : 200.0
FGSM Typical	ϵ : 0.3
FGSM Mild	ϵ : 0.05
Momentum Large	(nb : 10, ϵ : 16.0, ϵ_i 2.0)
Momentum Typical	(nb : 10, ϵ : 0.3, ϵ_i 0.06)
Momentum Mild	(nb : 10, ϵ : 0.15, ϵ_i 0.06)
Momentum V Mild	(nb : 10, ϵ : 0.1, ϵ_i 0.03)

ϵ is the size of the perturbation, ϵ_i is the change of the size each iteration and nb is the number of iterations.

C Supplementary Figures

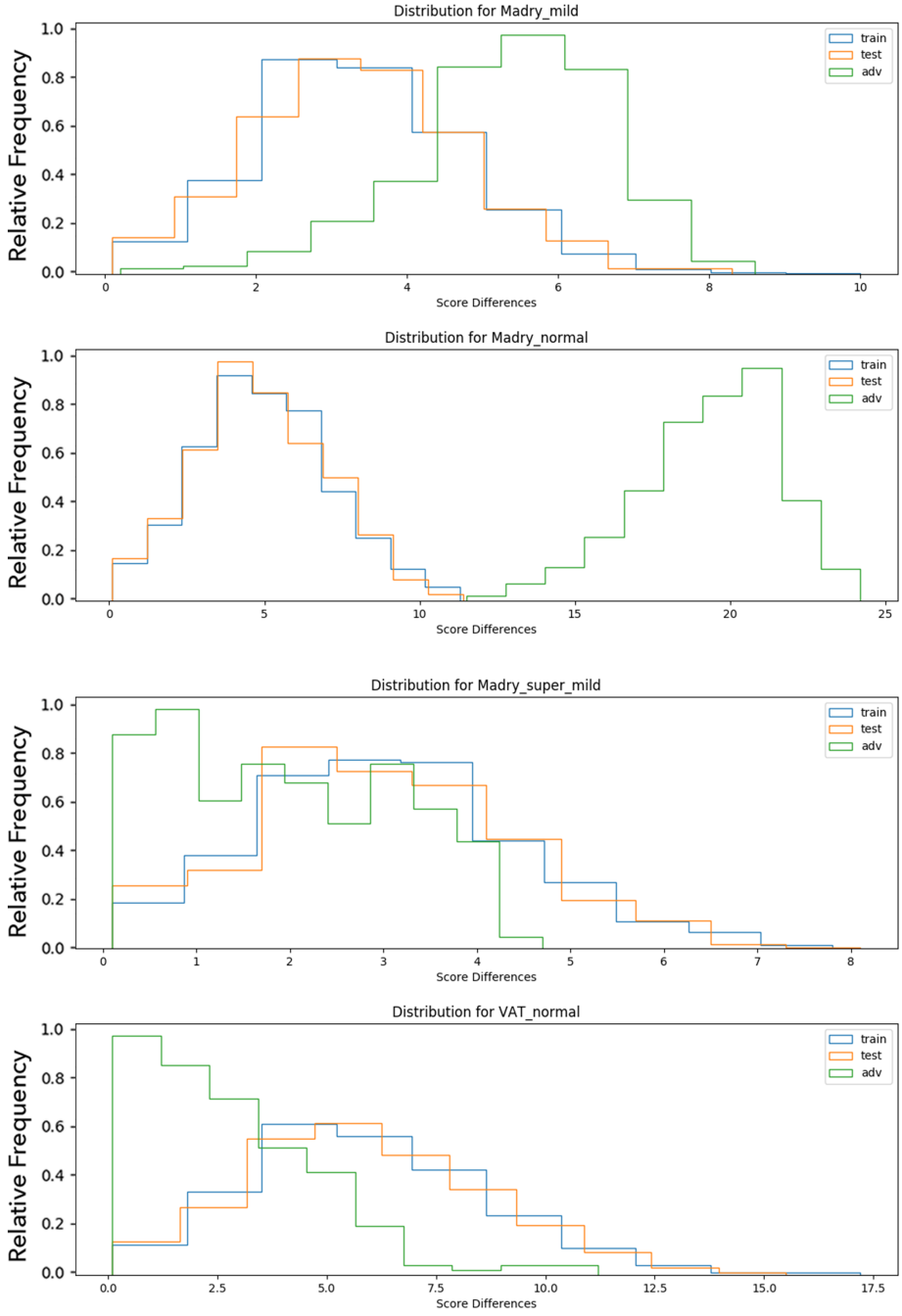


Figure 31: Extra distribution figures. These are the logit difference distributions figures for all attacks not in figures in the results section. They show the histograms of where the images sit in logit space, for adversarial sets in green, training sets in blue and test sets in orange. The figures show varying degrees of overlap in the histograms, which often correlate with the efficacy of the adversarial defence proposed in this thesis.

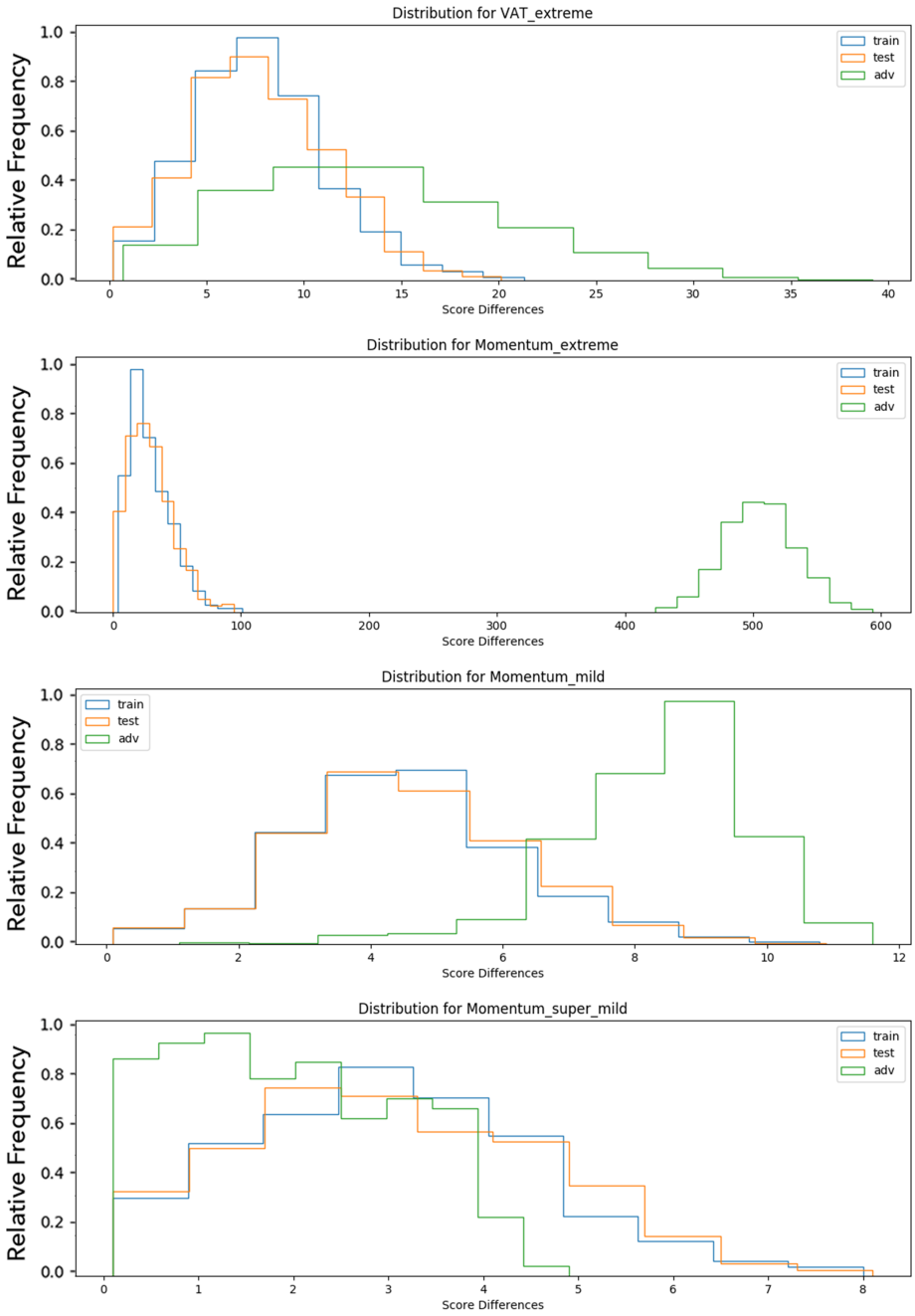


Figure 32: Extra distribution figures

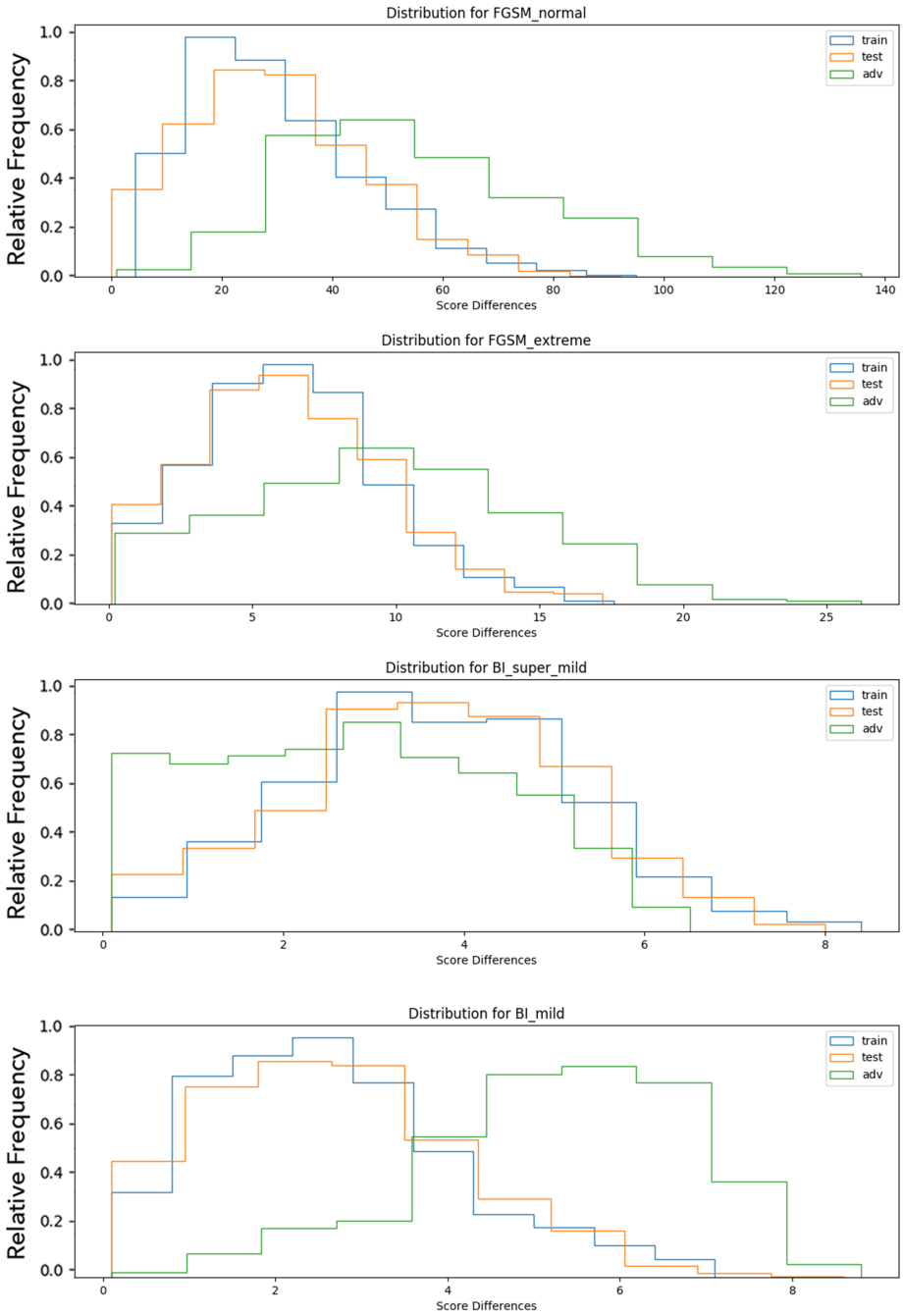


Figure 33: Extra distribution figures

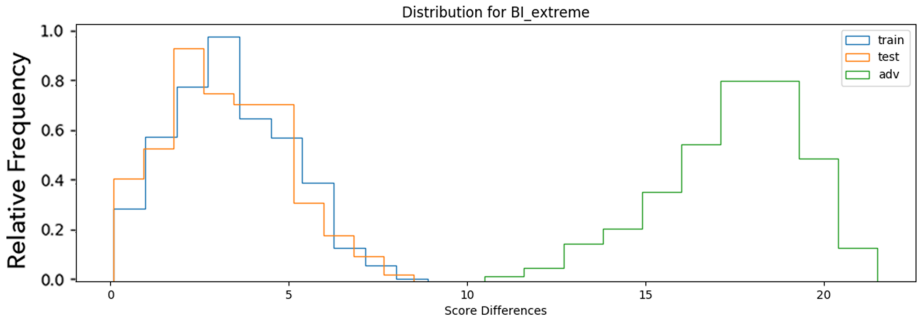


Figure 34: Extra distribution figures

References

[1] Martín Abadi et al. "TensorFlow: A System for Large-Scale Machine Learning." In: *OSDI*. Vol. 16. 2016, pp. 265–283.

[2] Miriam Ayer et al. "An empirical distribution function for sampling with incomplete information". In: *The annals of mathematical statistics* (1955), pp. 641–647.

[3] John Bradshaw, Alexander G de G Matthews, and Zoubin Ghahramani. "Adversarial examples, uncertainty, and transfer testing robustness in gaussian process hybrid deep networks". In: *arXiv preprint arXiv:1707.02476* (2017).

[4] John S Bridle. "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition". In: *Neurocomputing*. Springer, 1990, pp. 227–236.

[5] Nicholas Carlini and David Wagner. "Defensive distillation is not robust to adversarial examples". In: *arXiv preprint arXiv:1607.04311* (2016).

[6] Filipe Condessa, José Bioucas-Dias, and Jelena Kovačević. "Performance measures for classification systems with rejection". In: *Pattern Recognition* 63 (2017), pp. 437–450.

[7] Leda Cosmides and John Tooby. "Are humans good intuitive statisticians after all? Rethinking some conclusions from the literature on judgment under uncertainty". In: *cognition* 58.1 (1996), pp. 1–73.

[8] David R Cox. "The regression analysis of binary sequences". In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1958), pp. 215–242.

[9] Morris H DeGroot and Stephen E Fienberg. *Assessing Probability Assessors: Calibration and Refinement.* Tech. rep. CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF STATISTICS, 1981.

[10] Yinpeng Dong et al. "Boosting adversarial attacks with momentum". In: *arXiv preprint* (2018).

[11] Rob A Dunne and Norm A Campbell. "On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function". In: *Proc. 8th Aust. Conf. on the Neural Networks, Melbourne*. Vol. 181. 1997, p. 185.

[12] Andrew H Gee and Richard W Prager. “Limitations of neural networks for solving traveling salesman problems”. In: *IEEE Transactions on Neural Networks* 6.1 (1995), pp. 280–282.

[13] Georgia Gkioxari, Ross Girshick, and Jitendra Malik. “Contextual action recognition with r* cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1080–1088.

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

[15] Kathrin Grosse et al. “On the (statistical) detection of adversarial examples”. In: *arXiv preprint arXiv:1702.06280* (2017).

[16] Shixiang Gu and Luca Rigazio. “Towards deep neural network architectures robust to adversarial examples”. In: *arXiv preprint arXiv:1412.5068* (2014).

[17] Chuan Guo et al. “On calibration of modern neural networks”. In: *arXiv preprint arXiv:1706.04599* (2017).

[18] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. “Distilling the knowledge in a neural network”. In: *arXiv preprint arXiv:1503.02531* (2015).

[19] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural networks* 2.5 (1989), pp. 359–366.

[20] Gao Huang et al. “Densely connected convolutional networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. Vol. 1. 2. 2017, p. 3.

[21] Michael I Jordan et al. *Why the logistic function? A tutorial discussion on probabilities and neural networks*. 1995.

[22] Alex Kendall and Yarin Gal. “What uncertainties do we need in bayesian deep learning for computer vision?” In: *Advances in neural information processing systems*. 2017, pp. 5574–5584.

[23] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[24] Meelis Kull and Peter Flach. “Novel decompositions of proper scoring rules for classification: Score adjustment as precursor to calibration”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2015, pp. 68–85.

[25] Meelis Kull, Telmo M Silva Filho, Peter Flach, et al. “Beyond sigmoids: How to obtain well-calibrated probabilities from binary classifiers with beta calibration”. In: *Electronic Journal of Statistics* 11.2 (2017), pp. 5052–5080.

[26] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. “Adversarial examples in the physical world”. In: *arXiv preprint arXiv:1607.02533* (2016).

[27] Zachary C Lipton and Jacob Steinhardt. “Troubling Trends in Machine Learning Scholarship”. In: *arXiv preprint arXiv:1807.03341* (2018).

[28] Mark Lyksborg et al. “An Ensemble of 2D Convolutional Neural Networks for Tumor Segmentation”. In: *Image Analysis*. Ed. by Rasmus R. Paulsen and Kim S. Pedersen. Cham: Springer International Publishing, 2015, pp. 201–211. ISBN: 978-3-319-19665-7.

[29] Aleksander Madry et al. “Towards deep learning models resistant to adversarial attacks”. In: *arXiv preprint arXiv:1706.06083* (2017).

[30] Edward C Malthouse. “Limitations of nonlinear PCA as performed with generic neural networks”. In: *IEEE Transactions on neural networks* 9.1 (1998), pp. 165–173.

[31] Aditya Krishna Menon et al. “Predicting accurate probabilities with a ranking loss”. In: *Proceedings of the... International Conference on Machine Learning. International Conference on Machine Learning*. Vol. 2012. NIH Public Access. 2012, p. 703.

[32] Jan Hendrik Metzen et al. “On detecting adversarial perturbations”. In: *arXiv preprint arXiv:1702.04267* (2017).

[33] Takeru Miyato et al. “Distributional smoothing with virtual adversarial training”. In: *arXiv preprint arXiv:1507.00677* (2015).

[34] Seyed Mohsen Moosavi Dezfooli, Alhussein Fawzi, and Pascal Frossard. “Deepfool: a simple and accurate method to fool deep neural networks”. In: *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. EPFL-CONF-218057. 2016.

[35] Allan H Murphy and Robert L Winkler. “Reliability of subjective probability forecasts of precipitation and temperature”. In: *Applied Statistics* (1977), pp. 41–47.

[36] Radford M Neal. “Priors for infinite networks”. In: *Bayesian Learning for Neural Networks*. Springer, 1996, pp. 29–53.

[37] Andrew Y Ng and Michael I Jordan. “On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes”. In: *Advances in neural information processing systems*. 2002, pp. 841–848.

[38] Anh Nguyen, Jason Yosinski, and Jeff Clune. “Deep neural networks are easily fooled: High confidence predictions for unrecognizable images”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 427–436.

[39] Alexandru Niculescu-Mizil and Rich Caruana. “Predicting good probabilities with supervised learning”. In: *Proceedings of the 22nd international conference on Machine learning*. ACM. 2005, pp. 625–632.

[40] Nicolas Papernot, Nicholas Carlini, et al. “cleverhans v2. 0.0: an adversarial machine learning library”. In: *arXiv preprint arXiv:1610.00768* (2016).

[41] Nicolas Papernot, Patrick McDaniel, Somesh Jha, et al. “The limitations of deep learning in adversarial settings”. In: *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE. 2016, pp. 372–387.

[42] Nicolas Papernot, Patrick McDaniel, Xi Wu, et al. “Distillation as a defense to adversarial perturbations against deep neural networks”. In: *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE. 2016, pp. 582–597.

[43] Miquel Perello-Nieto et al. “Background Check: A general technique to build more reliable and versatile classifiers”. In: *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE. 2016, pp. 1143–1148.

[44] John Platt et al. “Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods”. In: *Advances in large margin classifiers* 10.3 (1999), pp. 61–74.

[45] Michael D Richard and Richard P Lippmann. “Neural network classifiers estimate Bayesian a posteriori probabilities”. In: *Neural computation* 3.4 (1991), pp. 461–483.

[46] Andras Rozsa, Ethan M Rudd, and Terrance E Boult. “Adversarial diversity and hard positive generation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2016, pp. 25–32.

[47] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. “Dynamic Routing Between Capsules”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 3856–3866. URL: `http://papers.nips.cc/paper/6975-dynamic-routing-between-capsules.pdf`.

[48] Mahmood Sharif et al. “Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2016, pp. 1528–1540.

[49] Christian Szegedy et al. “Intriguing properties of neural networks”. In: *arXiv preprint arXiv:1312.6199* (2013).

[50] Jordi Vallverdú. *Bayesians versus frequentists: A philosophical debate on statistical reasoning*. Springer, 2015.

[51] Bianca Zadrozny and Charles Elkan. “Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers”. In: *ICML*. Vol. 1. Citeseer. 2001, pp. 609–616.

[52] Bianca Zadrozny and Charles Elkan. “Transforming classifier scores into accurate multiclass probability estimates”. In: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2002, pp. 694–699.

[53] Langche Zeng. “Prediction and classification with neural network models”. In: *Sociological methods & research* 27.4 (1999), pp. 499–524.