Bachelor's Thesis Computing Science

Radboud University Nijmegen

Analysis and exploration of polynomial activation function: softRmax

---

A supposedly more robust alternative to softmax.

Author:
Nemo Ingendaa
s1063653

First supervisor/assessor:
dr. Marco Loog

Second assessor:
dr. Tom Claassen

June 19, 2024

Abstract

We analyze and extend research on the softRmax activation function, a claimed robust alternative to the traditional softmax function used in the final layer of neural networks. Through a series of experiments, we compare the performance of softRmax to softmax under various adversarial attacks, evaluating claims made in its introductory paper. Our results indicate that softRmax offers enhanced robustness against adversarial attacks conducted in a white-box setting, but does not improve the robustness of models against black-box attacks. However, for conclusive findings, further research is necessary.

Contents

Chapter 1

Introduction

The paper *Enhancing Classifier Conservativeness and Robustness by Polynomiality* [1] introduced an alternative to the widely used softmax activation function in the final layer of machine learning networks. The authors criticised the overconfident nature of softmax and proposed a more conservative alternative named 'softRmax'. They link softmax's behaviour to a lack of resilience against outliers in the data, which can have significant consequences in safety-critical systems such as autonomous driving or medical diagnosis.

The paper claims that substituting standard softmax with softRmax can significantly enhance the performance of classification models under adversarial attacks. Their claims are supported by experiments on various adversarial attack methods, which are designed to deceive classifiers by exploiting their vulnerabilities with carefully crafted data [2]. While they can be used by attackers to disrupt machine learning models, many of these attack methods are developed by researchers to assess the robustness of models. The term 'robustness' denotes a model's ability to deal with uncertainties and variance in the testing data.

In the domain of machine learning, adversarial robustness has been relatively underexplored compared to model architecture and training techniques. The original softRmax paper was the first to explicitly criticise the robustness aspect of softmax. Previous studies on softmax alternatives focused on critiquing the inefficiency of the activation function. Although efficiency is not directly relevant to the machine learning models in our paper, it can be of significant importance for large models such as (large) language models.

Attempts to address the inefficiency of softmax include alternative functions: sparsemax [3] and Taylor-softmax [4]. Sparsemax differs in its treatment of classes that the network assigns a very low likelihood, it takes a contrasting approach to softRmax by assigning a probability of zero to such classes. The paper on Taylor-softmax, similar to the original softRmax paper, criticizes the exponential nature of softmax. The authors behind Taylor-softmax argue it contributes to the inefficiency of the function, while softRmax's authors link it to the vulnerability to adversarial attacks.

The majority of research on enhancing model robustness is done on adversarial defence methods. The three prominent methods are adversarial training [5], defensive distillation [6], and gradient regularization. The last method utilises the controversial technique of gradient obfuscation, which 'gives a false sense of security' [7]. Our and the original research do not include a comparison with these defence methods. However, in the original paper is claimed that softRmax can 'outperform state-of-the-art adversarial defense approaches' in certain scenarios. Beyond performance, a strong argument for using softRmax is the ease of implementation and its minimal impact on a model's training time.

In the original research paper softRmax's impact on robustness is compared to that of softmax through experiments on a small variety of adversarial attacks performed on popular image classification datasets. In our research, we aim to validate the claims and results of the original paper by reproducing their experiments and extending them with two additional adversarial attacks: DeepFool [8] and Boundary Attack [9]. By aiming for minimal changes to the original data, these attacks are considered more accurate and reliable measurements of model robustness compared to those used in the original research.

Ensuring the reproducibility and reliability of findings is essential, particularly in the rapidly evolving field of machine learning where new techniques are frequently proposed. If the results of the original softRmax paper hold true and the improvements extend to a broader range of attacks, softRmax could be a valuable alternative to softmax in scenarios where robustness is of importance. Our research question is thus: *To what extent does softRmax position itself as a robust alternative to softmax for image classification tasks?*

The structure of the paper is as follows: Chapter 2 is dedicated to providing background knowledge for this thesis. In Chapter 3, we introduce SoftRmax, discussing the motivation behind it, its mathematical foundation, and the performance claims made by its authors. Chapter 4 details the methodology and setup of our research experiments, while Chapter 5 presents the outcomes of these experiments. Finally, Chapter 6 concludes the paper by discussing the limitations and findings of our research.

Chapter 2

Preliminaries

This chapter provides the theoretical foundation for our research, covering the essential concepts of activation functions and adversarial attacks necessary for understanding the experiments and analysis in this paper.

2.1 Output layer activation function

This section provides information on the output layer of a neural network, emphasizing the role of the activation function in this layer and highlighting the most commonly used function: softmax. Since this paper centres on this specific aspect of neural networks, we will not delve into the activation functions used in the hidden layer(s) of the network. While knowledge of the other aspects of the network is not necessary to get the main premise of this paper, it can help with understanding how the behaviour of the output layer activation function impacts the other components of a network.

The structure of a neural network is designed such that each node in the final hidden layer represents a single class. Thus, when training a network for a 10-class dataset, there should be 10 nodes in the last hidden layer. All nodes in this layer are then connected to one final node in the output layer, which consists of an activation function that collects the outputs of all classes and converts them into the desired format. For multi-class classification, this format is a probability distribution that sums to 1. This transformation is crucial for the training process, as it directly impacts an important component of the training process, the loss function.

After a network has made its predictions on a batch of training data, a loss function is used to evaluate how well its predictions align with the true labels of the data. In multi-class classification, the most commonly used loss function is the Logarithmic Loss, also known as Log Loss or Cross-Entropy Loss [10]. The loss function takes as its input, the vector of probabilities calculated by the activation function and outputs a score that reflects the model's error rate. Based on the loss function, the network is adjusted through a process known as backpropagation. Backpropagation uses the gradient of the loss function to

determine how the internal weights of the network need to be adjusted to minimise the overall loss and predict the data correctly the next time. For this paper, the details of backpropagation are not relevant, the key point to understand is that the activation function has a direct impact on the network's loss, and consequently on how the network is tuned during training.

The standard output layer activation function for multiclass classification is softmax. Softmax is a generalization of the sigmoid function, which is used for logistic regression. Both sigmoid and softmax transform an input into a probability between 0 and 1. However, the sigmoid function only takes a single value as input and outputs a single probability, which does not account for multiple classes. Using sigmoid on a vector of values will result in individual probabilities for each class, that don't sum to 1. In contrast, softmax takes a vector of values and converts them into a probability distribution that sums to 1. It achieves this by normalizing the output of each class, dividing the exponential of each input value by the sum of the exponentials of all input values

The formulas below illustrate how softmax (2.1) incorporates the term e^x ($= \frac{1}{e^{-x}}$) from the sigmoid function (2.2). The e in this term represents Euler's number, the base for natural logarithms. Although softmax can work with a different positive constant, e is chosen for its natural connection to growth processes and its convenience in differentiation, as the derivative of e^x with respect to x is e^x.

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \quad \text{for } i = 1, 2, \ldots, N \tag{2.1}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

Softmax has long been the default output layer activation function for multi-class classification tasks. However, this does not preclude the possibility that other activation functions could be used and improve accuracy, efficiency, or in our case, robustness. As we mentioned in the introduction, there has been research done on alternatives to softmax, but none have yet caused a major shift in practice.

2.2 Adversarial Attacks

In this section, we introduce the concept of adversarial attacks, focusing particularly on image classification tasks. Adversarial attacks aim to fool machine learning models by exploiting vulnerabilities in their decision-making process. In the context of image classification, the attacks work by modifying data with 'perturbations', noise that can vary significantly in their impact on a model. The success of an adversarial attack depends on various factors, such as its inherent strength, the complexity of the data, and possible defence methods used by the target model. Ultimately, success is measured by its capability to make a model misclassify images that it once correctly identified.

The security risks associated with adversarial attacks have raised concerns about the deployment of (deep) neural networks in sensitive areas. Enhancing the robustness of a model can help with countering attacks but also enhance a model's ability to correctly predict real-world outliers. For instance, consider an image recognition system in a self-driving car, which must be able to operate in a variety of circumstances, including scenarios with damaged road signs, faded lines, and non-ideal weather conditions. In the field of machine learning, the term 'robustness' is commonly used to describe a model's ability to maintain its performance when faced with uncertainties, including natural and adversarial outliers.

Adversarial attacks on neural networks can be broadly categorized into two primary types based on the level of access the attacker has to the model: white-box attacks and black-box attacks. To illustrate this distinction, we introduce the popular FGSM attack, which is one of the attacks used in the experiments of our research.

2.2.1 White box

One of the first proposed attacks for generating adversarial images was the FGSM attack, short for the 'fast gradient sign method'. It was introduced in the paper "Explaining and Harnessing Adversarial Examples" [2] in 2015. Despite its age, FGSM remains one of the most widely used attacks for assessing adversarial robustness due to its balance between ease of use, speed, and effectiveness.

The attack works by first computing the network loss after the original data has been fed into the network. Then the gradients are computed with respect to the pixels of the images. As the gradient of the loss function dictates the direction in which the weights should be adjusted, by reversing this direction, the FGSM attack maximises the loss. The resulting collections of pixel values are called noise or perturbation. The strength of the attack is controllable with parameter ϵ (epsilon). It is multiplied by the values of the noise and so controls the extent to which the original images are altered. An equation representing the FGSM attack is shown in Figure 2.1. In this equation, J denotes the loss function, where its inputs consist of the network parameters, the clean input image, and the true label of x respectively. ∇_x represents the gradient of x.

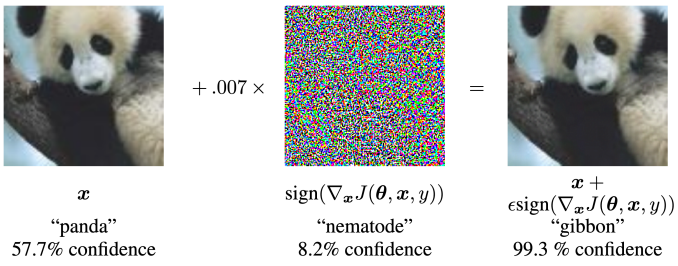


Figure 2.1: A demonstration of FGSM, where an image of a panda is misclassified by adding a small amount of noise.

As depicted in Figure 2.1, FGSM calculates the sign of the gradient before adding it to the original pixel values. This sign function assigns positive values to 1 and negative values to -1, resulting in pixels being perturbed by the same amount regardless of their original values, whether it's 0.1 or 5. While this approach might seem counterintuitive, the motivation behind it, as stated in the paper behind FGSM[2], is to achieve misclassification with modifications that are 'small enough' to remain unnoticed. Using the full gradient information would cause larger changes to the input.

Standard FGSM is a white-box attack, meaning that the attacker has full access to the model and can, in our case, compute the gradient of the loss. In contrast, in a black-box environment, the attacker has limited access to information about the model. The black-box attacks discussed in this paper will only have the possibility to query for output labels. The attacker can submit inputs to the model and utilise the probability distribution it returns but does not know the model's parameters or architecture.

2.2.2 Black box

Even though FGSM requires access to a model's internal information, a black-box version can be constructed using a 'donor network'. An attacker can train its own model on the same or a similar dataset as the target model, such that the adversarial samples it creates will have a comparable effect on the target model. However, creating a donor network that matches the target model can be challenging, requiring similar network architecture and training parameters. In our research, alongside the black-box version of FGSM, models will be subjected to black-box attacks that do not require the use of a donor network. These attacks generate perturbations without the need for any model information.

While much of the research on adversarial attacks in image classification is centered around the white-box setting, black-box attacks hold greater relevance in real-world scenarios. Constructing black-box attacks is more challenging, but they are also more difficult to defend against. Each type of attack demands distinct defence strategies. As white-box attacks exploit model information, most of the adversarial defence methods focus on obscuring this information, whereas defending against black-box attacks requires a broader sense of robustness. That is why the experiments in this paper will feature a variety of adversarial attacks. The next chapter delves into how softRmax aims to provide defence against both white-box and black-box attacks.

Chapter 3

Introduction of SoftRmax

In this chapter, we examine the key differences between softmax and softRmax, outlining the authors' motivation behind the creation of softRmax as a robust alternative to softmax. Additionally, we examine the mathematical foundations of these functions and use simple examples to illustrate their behaviour and give us an understanding of their impact on deep neural networks (DNNs).

3.1 Motivation behind softRmax

The authors behind softRmax critiqued the use of softmax due to concerns about deep neural networks being vulnerable to outliers and overfitting. They attribute these issues to the tendency of these models to make overconfident predictions at the tail ends of the distribution, representing samples that deviate from the training data (the bulk of the distribution). They state: 'A model should not be certain about a sample that deviates too much from the training data. Overconfident predictions on samples in the distribution tails should often be avoided' [1]. To express this concept, the authors use the real-world example of an atypical patient being classified as either healthy or diseased with high confidence, illustrating the potential risks of making overconfident predictions about outliers.

The authors attribute this undesirable behaviour to the exponential form of the softmax function. With softRmax, they strive towards 'conservativeness', predicting samples in the tail of the distribution close to random guess-level. This is achieved by shifting from exponential to polynomial behaviour. The mathematics behind this change will be discussed later on in Section 3.3.

The conservative approach of the softRmax activation function is one of two aspects contributing to its improved adversarial robustness. The second aspect is its 'inherent gradient regularization', which leads to an enlarged margin between samples and the decision boundary. Both aspects are illustrated in Figure 3.1.

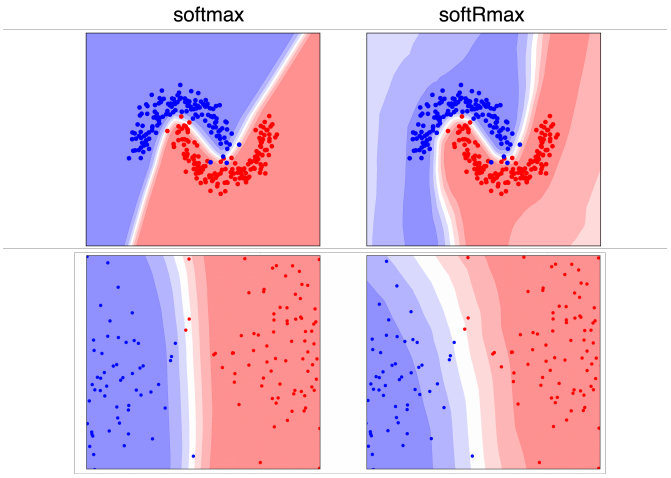


Figure 3.1: Visualisation that illustrates the difference in posterior probability (confidence) between softmax and softRmax for a simple two-class dataset. It highlights the two factors of softRmax that are claimed to enhance robustness: conservativeness (top) and enlarged margin (bottom). *Colour saturation indicates the level of confidence.*

The decision boundary is the region that separates different data classes, where the classifier is uncertain about its prediction. During the training phase of a neural network, the boundary changes in response to the loss function. Generally, there are two ways for a network to deal with data that is positioned close to the decision boundary.

According to the paper, softmax-based models mostly manage these data samples by making the 'posterior transient' steep at the decision boundary. This results in the model predicting samples in the area close to the absolute middle of the boundary with higher confidence. The posterior probability (confidence) between two classes transitions from close to 100% for class A to 50% and then back to almost 100% for class B. The rate at which the confidence shifts when moving towards or away from the boundary is defined as the steepness of the decision boundary. Softmax models tend to increase the steepness, causing outlying samples near the boundary to be predicted with higher confidence. This difference in behaviour towards the decision boundary is visualised in the bottom plots of Figure 3.1.

The original paper notes that while softmax's approach towards the decision boundary helps a model accurately separate the data into classes, it also makes it more vulnerable to adversarial attacks. The reasoning is that once the boundary has been adjusted based on the initial outliers, in the subsequent epochs it will undergo minimal change, limiting training optimization. As mentioned earlier, softRmax approaches these outliers differently. Rather than confidently assigning the outliers to a specific class, a softRmax model will reduce its confidence for these samples. While the samples will still be correctly classified by the model, they will have a smaller impact on the overall learning direction of the model. Which should consequently reduce the risk of overfitting and enhance adversarial robustness.

3.2 Performance claims

In this section, we outline the claims made in the softRmax introductory paper regarding the robustness gains a deep learning model could achieve by substituting softmax for softRmax. The main premise of the paper, repeated three times, is: 'We demonstrate that a higher robustness of neural networks can be obtained by simply substituting the standard softmax with our softRmax'. This emphasizes the ease of replacing softmax with softRmax, a factor that could be beneficial when compared to other adversarial defence methods.

The paper emphasizes that softRmax's enhanced robustness does not rely on gradient obfuscation, a technique frequently utilized by defence methods belonging to the gradient regularization category. Gradient obfuscation is criticised for hindering gradient-based attacks by creating an 'unnecessarily rough loss landscape' [1]. It achieves this by incorporating a 'penalty term' into the loss function, which makes sure the loss function will not return large gradients. Relying on gradient obfuscation is considered bad because it does not solve the core issue of adversarial attacks. Gradient obfuscation is generally ineffective against high iteration attacks such as BIM, and black-box attacks. To demonstrate that softRmax does not depend on gradient obfuscation, the softRmax paper conducts experiments involving five different attacks, including two black-box attacks.

Their experiments show that softRmax models exhibit considerably better adversarial performance across all attacks. The improvements range from doubling the prediction accuracy for weaker attacks, to maintaining accuracy well above random guessing levels in scenarios where an attack would reduce a standard softmax model's accuracy below this threshold. Their results show comparable improvements for both black-box and white-box attacks, which is a promising prospect.

3.3 Technical analysis

This section will not include the full mathematical background and thought process behind softRmax, this is all covered extensively in the original paper[1]. Instead, we discuss the most relevant aspects of the function and compare them to softmax. For this comparison the visualisations in Figure 3.3 will function as a helping asset. This figure shows plots of softmax and softRmax for 2 classes, with the x-axis and y-axis representing the input values, and the z-axis indicating the posterior probability of the first class.

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \quad \text{for } i = 1, 2, \ldots, N \tag{3.1}$$

$$\sigma(z)_i = \frac{\frac{1}{\|z-e_i\|^2}}{\sum_{j=1}^{N} \frac{1}{\|z-e_j\|^2}} \quad \text{for } i = 1, 2, \ldots, N \tag{3.2}$$

The basis of the formulas for softmax (3.1) and softRmax (3.2) is logically the same, as both use normalisation to output a probability distribution. The difference between the functions is the term representing the likelihood of a class. We compare e^{z_i} with $\frac{1}{\|z-e_i\|^2}$. It is important to first off mention that the e in both formulas represents something different. As explained in Section 2.1, e represents Euler's number in the formula of softmax. For softRmax however, e_i represents the ith column of the identity matrix. The identity matrix is a $N \times N$ square matrix with the diagonal elements equal to 1 and all other elements equal to 0, N represents the number of classes. The formula computes the squared Euclidean distance between a row of the matrix and the logits vector z. By then taking the inverse of the outcome, the class with the smallest Euclidean distance to z gets the biggest value for numerator of the formula. The class consequently, receives the highest probability, as the denominator is also a fraction smaller than zero. It is important to note that the class order remains unchanged, regardless of whether softRmax or softmax is used as the activation function. The difference between their outputs is the probabilities they assign. To illustrate this, let's consider some examples.

Figure 3.2 shows example output vectors for a three-class classification, followed by the probability distributions after conversion through softmax and softRmax. These examples were chosen to highlight the distinctive characteristics of the two functions. The leftmost vector includes values that are quite close together, simulating a network that is uncertain about its prediction. For this vector, softRmax assigns a higher posterior probability to the third class than softmax does. This behaviour is visible in Figure 3.3, which shows that the slope of the softRmax function is very steep between 0 and 1. This makes its output overly sensitive to small input differences within that range. While the slope of the softmax function is also at its steepest around 0, compared to softRmax its confidence rises more gradually.

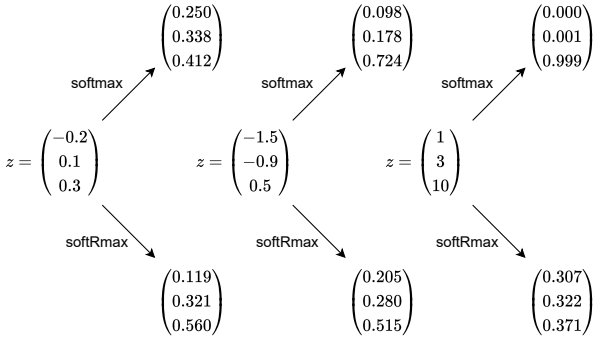


Figure 3.2: Confidence comparison between softmax and softRmax on example outputs

The two other examples in Figure 3.2 highlight the conservative tails of softRmax. The values of the input vectors are further apart, simulating the classification model being more confident about its prediction. SoftRmax converts the probability of all classes towards random guessing. While for softmax the probability of the third class approaches 100%.

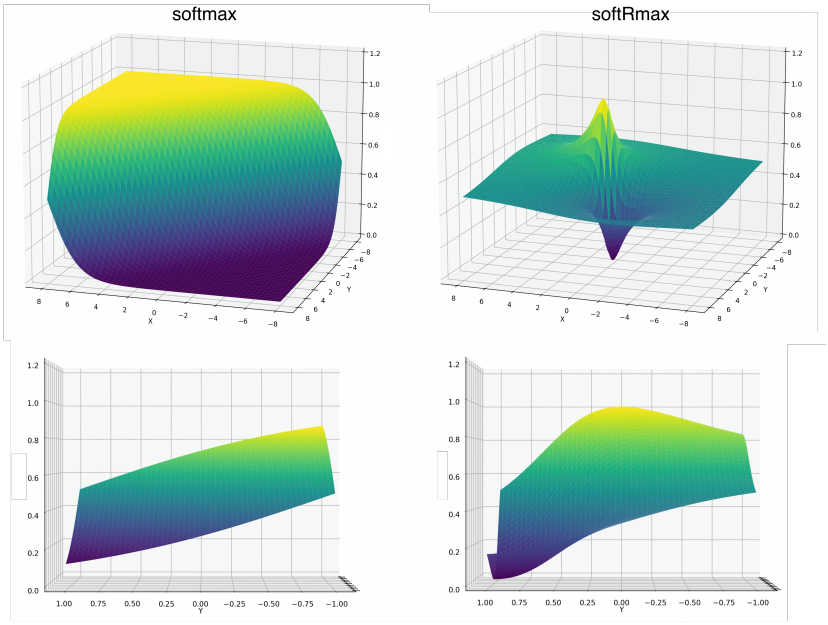
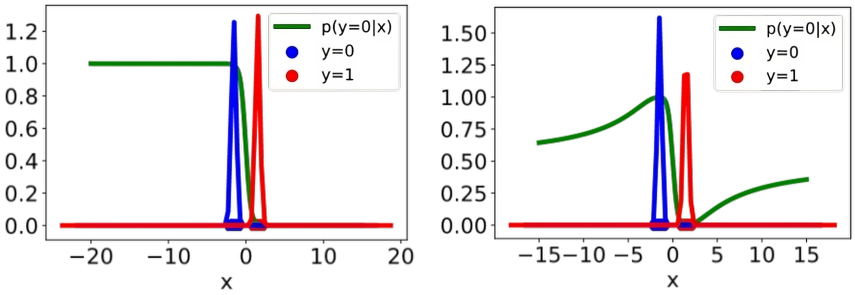


Figure 3.3: Visual comparison between the softmax and softRmax functions for 2 classes. In the plots, the x-axis and y-axis represent a class, while the z-axis shows the posterior probability for the class on the x-axis. In the top softRmax plot, the posterior probability initially rises sharply towards 100% before slowly declining toward 50%, suggesting more conservative predictions for samples in the tail. Conversely, in the softmax plot, the posterior probability gradually increases towards 100%. The bottom plots offer a detailed view of the functions in the range of -1 to 1.

Having compared the formula and behaviour of softRmax with the standard softmax, we now discuss the development of softRmax as an alternative activation function. As outlined in Section 3.1, the motivation for creating an alternative to softmax stemmed from a critique of its exponential behaviour, which we demonstrated with the examples from Figure 3.2. The paper links softmax's behaviour to that of the Gaussian distribution in Linear discriminant analysis (LDA), noting its exponentiality away from the mean of the data. The distribution is compared to a polynomial counterpart, the standard Cauchy distribution, which features heavier tails compared to the Gaussian distribution.

In the original paper a simple 1D classification experiment is conducted, with two uniform distributions representing the data classes. The Gaussian and Cauchy distributions are fitted to the data using maximum likelihood to estimate the class probabilities. In 3.4 the green line in plots a and b shows the posterior probability for the blue class. The researchers drew inspiration from the conservative predictions for the samples in the tail of the Cauchy distribution. This inspiration is evident when comparing the shape of the posterior probability in plot b to that of softRmax in Figure 3.3.



(a) Gaussian distribution. (b) Student's t-distribution.

Figure 3.4: LDA with exponential and polynomial assumptions. Posteriors of class $y = 1$ are compared. Plots a and b show the predicted posterior by LDA with class conditionals Gaussian and Cauchy (student's t). The Gaussian distribution shows overconfident predictions in the distribution tails. While Cauchy shows conservative prediction, achieved with polynomiality. From *Enhancing Classifier Conservativeness and Robustness by Polynomiality* [1]

However, while softRmax draws inspiration from the polynomiality of the Cauchy distribution, its use of the Euclidean distance and the identity matrix is not derived from the distribution's formula: $\frac{1}{\pi(1+x^2)}$. The original paper does not clarify the role of these two elements in softRmax. Therefore, we suspect that the Euclidean distance and the identity matrix do not play a direct role in the enhanced robustness of the function.

Chapter 4

Experiments

In this chapter, we delve into the research experiments, detailing the setup and the reasoning behind them. Our objective is to assess how softmax and softRmax differ in their impact on a model's adversarial robustness. To achieve this, we outline experiments that involve a range of adversarial attack methods applied to various widely used datasets.

4.1 Evaluating adversarial robustness

Assessing a model's robustness through adversarial attacks poses a challenge, given the vast selection of attack methods, each with its unique characteristics that exploit different weaknesses in the model. Our research will cover, in addition to a reproduction of the softRmax experiments, two additional adversarial attacks. These attacks are considered more powerful and aim to further elaborate on the aspects examined in the reproduction experiments.

In this context, a 'powerful' attack means more than its ability to reduce the prediction accuracy of a model towards 0%. Misclassification can be easily achieved by adding substantial noise to an image, often making it unrecognizable to both the model and a human observer. Therefore, we evaluate the powerfulness of an adversarial attack by the balance between the reduction in accuracy and the amount of noise it adds to the samples. In our assessment, a powerful attack should have the ability to deceive a classifier by changing the data as minimally as possible. This understanding drives the motivation behind the new experiments covered in Section 4.3.

The choice behind the adversarial attacks covered in our and the original research comes from the aspects that are said to make a softRmax model more robust. Gradient-based attacks naturally perturb a sample towards the direction where the overall loss increases as explained in Section 2.2.1 with FGSM. According to the original paper, the gradient from a softRmax model will have the attack shift the data samples towards the conservative tail of its distribution. Opposed to a model trained with softmax which will have the samples move in the direction of the class most dissimilar to the sample.

The second aspect of robustness: the enlarged margin, is evaluated with targeted attacks. These attacks ensure that the samples do not move towards the tail of the distribution by directing them to a target class instead, thereby putting the focus on the decision boundary between the classes.

Logically, the experimental chapter is divided into two parts. The experiments covered in the first part are a slight extension of the experiments performed in the softRmax paper. Meanwhile, the second part introduces new experiments that challenge softRmax's robustness aspects with attacks specially designed to maximise the impact on model robustness with minimal alteration to the original samples. The implementation of the experiments performed in this thesis can be found on GitHub [11].

4.2 Reproduction experiments

This section outlines the setup from the reproduction of the experiments conducted in the original paper. Our version extends the original research with a new dataset: Fashion-MNIST, which bridges the gap in complexity between the original two datasets. It will be subjected to the same adversarial attacks as the other datasets.

4.2.1 Experimental setup

This section provides an in-depth look at the experimental setup, encompassing the selection of datasets and network architectures. We discuss changes made compared to the original research, driven by unclear or missing information in the original paper, and challenges encountered during training.

4.2.1.1 Datasets

The original experiments were conducted on two widely used image classification datasets: MNIST and CIFAR. MNIST is among the least complex datasets, consisting of hand-drawn images of the digits 0 through 9, organized into 10 classes. Additionally, a two-class variation of MNIST is tested, known as MNIST 3&7, which is a subset of the original dataset that includes only classes 3 and 7.

Compared to MNIST, the CIFAR dataset poses a more challenging image classification task because it features colored images instead of grayscale and contains more complex images from categories like animals, vehicles, and trees. CIFAR comes in two versions: CIFAR10 and CIFAR100 While CIFAR10 like MNIST contains 10 classes, CIFAR-100 includes 10 times that amount. This makes CIFAR-100 notably more difficult to train a model on.

Noticing the significant complexity gap between CIFAR and MNIST, we introduce Fashion-MNIST as an intermediary dataset, it is for the remainder of the paper shortened to 'Fashion'. Although its images have the same 28 $times$28 grayscale format as those in MNIST, the images contain objects that present greater challenges for classifiers due to the subtle distinctions between its classes. It features fashion products from 10 categories.

For all datasets, we first standardise the samples before training the model. This commonly used pre-processing technique rescales the pixel values of the samples to fit a standard normal distribution. This step generally helps stabilize the training process, ensuring more consistent results [12]. Although standardisation is not explicitly mentioned in the softRmax paper, the pre-processing step is present in their GitHub implementation [13].

4.2.1.2 Network architectures and parameters

For MNIST and CIFAR, we will be using the network architectures that are listed in the softRmax paper. As for Fashion, we adopt a well-performing architecture from Kaggle [14]. Our setup of CIFAR10 slightly deviates from the original study, which performs the experiments on a VGG-16 model that is trained from the ground up. For us, this approach resulted in significant training problems, prompting us to opt for a pre-trained VGG-16 model. In Section 6.2, we discuss the implications of this modification and its effect on the experiment's validity.

The networks are trained on all training samples from their respective datasets, loaded with Torchvision [15]. The MNIST and Fashion datasets consist of 60,000 training samples each, while the CIFAR datasets both contain 50,000 samples. The experiments are conducted on distinct testing sets, that do not include images also present in the training sets. For all datasets, the testing set contains 10,000 samples.

An outline of the network architectures and training parameters is provided below in Table 4.1. The only difference in the training process between a model using softmax or softRmax is the loss function as explained in Section 3.2.

	MNIST	**Fashion**	**CIFAR10**	**CIFAR100**
Architecture	4 Conv + 1 Full	2 Conv + 3 Full	VGG16 [16]	RESNET-50 [17]
Optimizer	Adam	Adam	SGD	Adam
Epochs	5	20	15	15
Batch size	32	128	256	512
Learning rate	1e-3	2e-3	5e-3	1e-4
Weight decay	5e-6	5e-6	5e-6	5e-6
softmax ACC	98,95%	97,6%	85,27%	57,67%
softRmax ACC	99,14%	94,61%	84,69%	56,70%

Conv = convolutional layer, Full = fully connected layer, ACC = accuracy (clean data)

Table 4.1: Outline of the network architectures and parameters for the four datasets

4.2.2 Adversarial attacks

We divide the reproduction experiments into two categories: targeted and non-targeted attacks. The non-targeted experiments are performed with the FGSM and BIM attacks. The FGSM was explained in Section 2.2, which also covered the desitinction between between white-box and black-box attacks. Basic Iterative Method (BIM) is an extension of FGSM that performs the attack multiple times instead of once. BIM is deemed more powerful than FGSM, as it has a greater effect on model performance with the same value for epsilon.

The targeted attack experiments feature a variant of standard (non-targeted) FGSM in addition to the 'average-sample attack', that was proposed in the softRmax paper. Originally the targeted attacks were only performed on the MNIST dataset, but we extend it with Fashion and CIFAR.

As mentioned in the introduction, there is no consensus on how to evaluate the robustness of a model. However, for the experiments in this section, the most appropriate performance metric for a model under adversarial attack is the decrease in prediction accuracy between the clean and adversarial samples. Accuracy is defined as the number of correctly predicted samples divided by the total number of samples.

4.2.2.1 Non-targeted (standard) attacks

The first batch of experiments involving FGSM and BIM, we conduct in the same manner as the original research. In all cases, BIM is performed as 10 iterations of FGSM. Adversarial examples are generated for all test images using a trained softmax or softRmax model along with their respective loss function. Then that same model is evaluated on the new adversarial images and its prediction accuracy is calculated.

We repeat this process for a range of epsilon values, spanning from 0.1 to 0.5 with increments of 0.1 for the greyscale datasets. While the softRmax paper only lists accuracies for epsilon values of 0.1 and 0.3, we expand our testing to include a wider range of values to get a more comprehensive understanding of the decline in performance across different epsilon levels. For both CIFAR datasets, the attacks are inherently stronger, so we opt to use the lower epsilon values of 0.01 and 0.05 in addition to 0.1, and 0.3.

4.2.2.2 Targeted attacks

For the targeted attack experiments, we employ two attack methods: a gradient-based attack and a 'semi-black box' attack named the average-sample attack. While the softRmax paper does not specify which gradient-based attack is used, we believe it to be a modified version of FGSM. This attack is conducted in two settings: white-box and black-box.

The targeted version of FGSM differs from its standard form in the way the noise is constructed. In the conventional form, the loss is determined based on the difference between the predicted and true labels. However, in the targeted attack, the true labels are replaced with a single target class. That way the loss, and in turn the gradient, are focused more on the features that are relevant for the target class. By reversing

the direction of the perturbation, the samples are shifted towards the target class. This approach emphasizes the robustness provided by the extended margin more than the conservative tail.

The black box variant of this attack creates adversarial images based on the loss of a substitute network, as was discussed in Section 2.2.2. This makes it a weaker attack than its white box counterpart. The paper does not specify the activation function used in the substitute network. However, their implementation included another activation function, besides softRmax, called 'conservative softmax monotone' which we opted for this task. Additionally, the softRmax paper does not disclose the value of epsilon used in these experiments. So, we determined this based on the accuracy results from the non-targeted attacks. We settled on a value of 0.5 for MNIST and 0.3 for the other datasets.

The motivation given in the softRmax paper behind evaluating a black-box variant of the FGSM attack is to demonstrate that the robustness of softRmax does not depend on gradient obfuscation. As we mentioned in Section 3.2, this distinguishes softRmax from adversarial defence methods such as gradient regularisation.

In addition to the targeted FGSM attacks, the softRmax paper introduces a 'semi-black box attack' coined the average-sample attack (4.1). Why the authors categorise the attack as 'semi-black box' instead of black-box is not explained in the paper. We question the use of this term because the attack requires zero knowledge about the model. The idea of a semi-black box or grey box attack is that limited information about the model is used. The average-sample attack does not use the gradient of a network for perturbing a sample into a specific direction, it 'simply perturbs a sample to the direction of a selected target class based on a pre-computed average' [1]. In practice, this means that the attack calculates the average image of two classes, the adversarial and the target class. The difference between the average image of the target class and the average image of the adversarial class then becomes the perturbation that will be added to the original images. Just like with FGSM, we take the sign of the perturbation before adding it to the original images and control the attack's strength with the epsilon parameter. For consistency, the attack is performed on the same epsilon value as the targeted FGSM attacks.

$$x'_y = x_y + \epsilon \,\mathrm{sign}(\mathrm{Avg}_t - \mathrm{Avg}_y) \tag{4.1}$$

For the experiments in this section, we first calculate the results per class and later on average those percentages to get the accuracy we will present. The accuracy of a class is determined by computing the average accuracy of the class across instances where the other classes act as the target class. For example, the result of the fourth class will be the average of 9 iterations, each performed with a different target class.

4.3 New experiments

This section outlines the setup of the newly constructed experiments aimed at further substantiating our analysis of the robustness of softRmax. The objective of these experiments is not to explore new aspects of softRmax's robustness but rather to build on the previous experiments, particularly the targeted attacks.

4.3.1 Experimental setup

For the new experiments, we employ the same experimental setup utilised in the reproduction experiments, see Table 4.1. This includes the same trained models and data samples. However, due to the complexity of these new attacks, the generation of perturbations requires considerably more resources compared to the relatively simple attacks used in the first part. As a result, we had to make certain sacrifices to our experiments.

4.3.2 Adversarial attacks

The new experiments feature a gradient-based attack called DeepFool [8] and a decision-based attack named the Boundary Attack [9]. While both are different in the way they construct an adversarial image, their overall goal is the same. Both attacks aim to create a perturbation that just slightly pushes an image over the decision boundary. However, their starting points for this task are the opposite of each other. DeepFool starts with the original image and tries to find the minimal amount of noise that makes the model predict a different class. Meanwhile, the Boundary Attack starts with a synthetic image far from the original class and then slowly brings in more of the original image until the model predicts the right class.

Since both attacks aim to shift adversarial samples across the decision boundary, they serve as effective tools for evaluating softRmax's enlarged margin aspect. While targeted FGSM and average-sample attacks also explore this aspect, the new attacks offer more valuable insights by precisely positioning samples near the decision boundary of their original class while still causing misclassification. In contrast, the targeted attacks will require significant fine-tuning for each specific sample to achieve similar results. The way we perform those experiments with a fixed epsilon value, some samples may be positioned close to the decision boundary, but the majority will not even be near it.

As mentioned previously, we had to make sacrifices regarding which attacks we test on which datasets and to what extent. For both CIFAR variants, the amount of test samples for DeepFool is reduced and we were unable to complete valuable tests on the Boundary Attack. The exact number of data samples we use for each combination of attack and dataset is listed in Table 4.2.

	MNIST	Fashion	CIFAR10	CIFAR100
DeepFool	10.000	10.000	1.000	300
Boundary	500	500	N/A	N/A

Table 4.2: The number of data samples used per attack per dataset

4.3.2.1 Evaluating methods

For the two new adversarial attacks, we can not evaluate the model performance with prediction accuracy as both attacks iterate until all samples are converted, resulting in accuracies close to 0%, depending on the iteration limit set. That is why we have to assess the robustness directly from the images by measuring the disparity between the adversarial image and the original. For this, we use two functions (4.2 and 4.3) that calculate the differences between two images in a pixel-by-pixel way. One of the functions stems from the paper on DeepFool [8] and is accessible in a Python implementation of the DeepFool attack [18]. The second function is the mean squared error (MSE). The output of both functions is a strictly positive float; a higher output indicates a greater disparity between the input images.

We assess the difference in robustness between the two models by comparing their average scores on the same batch of samples. A model is deemed more robust when the adversarial images created for it deviate more from the originals. In other words, the more robust model will require larger pertubations added to the images to be fooled.

$$\frac{1}{|D|} \sum_{\boldsymbol{x} \in D} \frac{\|\hat{\boldsymbol{r}}(\boldsymbol{x})\|_2}{\|\boldsymbol{x}\|_2} \tag{4.2}$$

$$\frac{1}{|D|} \sum_{\boldsymbol{x} \in D} \left(\hat{\boldsymbol{r}}(\boldsymbol{x})\right)^2 \tag{4.3}$$

$$\hat{\boldsymbol{r}}(\boldsymbol{x}) := \hat{\boldsymbol{x}} - \boldsymbol{x} \text{ (perturbation)}$$

In the functions above, D represents the set of clean data samples, and $\hat{\boldsymbol{x}}$ denotes the adversarial image constructed for image $\boldsymbol{x}$. The formula for the DeepFool measurement is shown on top, with the mean squared error depicted below it.

Since both attacks iterate until a sample converges, it is impractical to include images that the model already predicts incorrectly before undergoing an attack. Including those images would skew the robustness measurements because the adversarial image would remain unchanged from its original state. Therefore, we first remove the incorrectly predicted samples from our dataset.

4.3.2.2 DeepFool

The DeepFool adversarial attack shares similarities with other gradient-based attacks, such as FGSM, in the way it generates perturbations. Like FGSM, DeepFool computes the gradient of the model's output with respect to the input images to determine the direction of perturbation. However, DeepFool differs from FGSM and other gradient-based attacks in several key aspects.

The main difference lies in how DeepFool computes the perturbation direction. DeepFool achieves finding small but effective perturbations by iteratively computing the difference between the model's current prediction and the closest class. It then steers the input image in the direction that reduces this difference.

Another distinction is that DeepFool operates under the assumption that the decision boundary of a model around the input image is a relatively straight line. It uses this assumption to efficiently compute the perturbation direction. However, we hypothesise that this approach may pose challenges for classifiers with more complex decision boundaries. Such classifiers may require DeepFool to undergo additional iterations or generate larger perturbations. Along with the nature of DeepFool steering samples toward the decision boundary of another class, it positions itself as a suitable attack for evaluating the enlarged margin and general shape of the decision boundary.

For DeepFool, we adhere to the default settings provided by the creator of the DeepFool implementation [18] we use for our experiments. Those settings proved to be effective and caused no issues.

An important practical point of notice is that the attack requires the raw output from the network without passing it through a final layer activation function. Consequently, in this experiment, the gradient is not directly influenced by the softRmax or softmax function. However, as the activation function does impact the direction of internal weights during training it, in turn, affects the gradient.

4.3.2.3 Boundary Attack

The boundary attack was introduced in its paper as the first 'effective decision-based attack that scales to natural datasets such as ImageNet and is applicable to deep neural networks (DNNs)' [9]. Decision-based attacks rely solely on the final output of the model and thus the Boundary Attack is a black box attack. It differentiates itself, like DeepFool, from the previously covered attacks by finding the minimum perturbation. However, the number of iterations for the Boundary Attack is significantly higher compared to DeepFool. While in our experience DeepFool uses on average 14 iterations to create its adversarial sample, the Boundary Attack requires at least 45,000. This substantial difference in iteration count arises from the fact that the Boundary Attack performs much smaller adjustments to the images during each iteration, ultimately resulting in adversarial images that are closer to the originals.

While the Boundary Attack evaluates the same robustness aspect as DeepFool, namely the enlarged margin, its addition to our experiment pool remains valuable. It helps us validate the claim of softRmax not relying on gradient obfustication along with the previous black-box attacks from Section 4.2.

The implementation of the Boundary Attack used in the experiments is sourced from the foolbox library [19]. Their implementation did require some modifications for it to work with our neural networks. Our modified version of the Boundary Attack file, along with documentation, is available in the 'extra' folder of this paper's GitHub repository [11]. We kept the majority of attack parameters to the default, except for the maximum number of steps (iterations), as the default amount was too low for most of our samples to converge.

Our attempts to conduct experiments with the Boundary Attack on both CIFAR datasets were unsuccessful, as indicated in table 4.2. The colored CIFAR images require a significantly higher number of iterations to converge compared to other datasets, coupled with longer processing times per iteration. Despite experimenting with different parameters, we were unable to overcome these challenges.

As mentioned earlier, the Boundary Attack operates in the opposite direction of DeepFool. It starts with a synthetic image and gradually moves it toward the original image along the decision boundary of the classifier, ensuring that it remains misclassified. The attack continues to adjust the sample until it reaches a point where further perturbations would cause it to cross the decision boundary of the true class or move it away from the original image. A high-level visualising of the Boundary Attack is shown in figure 4.1.

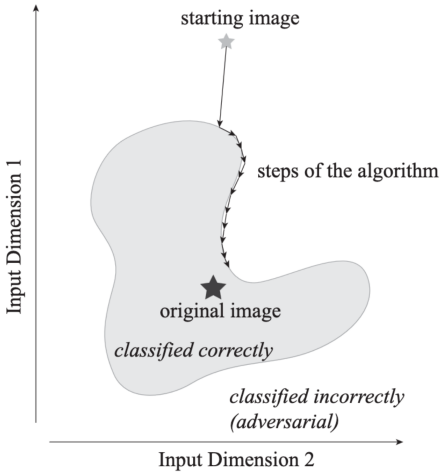


Figure 4.1: Basic intuition of the Boundary Attack. From *Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models* [9]

.

Chapter 5

Results

In this chapter, we delve into the outcomes of the experiments. The chapter follows the same structure as the previous experiments chapter, with two main sections: the reproduction experiments and the new experiments constructed by us. The detailed raw results can be found on the GitHub page [13].

5.1 Reproduction experiments

Our results of the reproduction experiments will be presented alongside those from the softRmax paper, allowing us to compare the robustness of our softRmax and softmax models while validating the paper's results.

5.1.1 Non-targeted (standard)

The Tables in 5.1 show our results next to those of the paper for the two epsilon values tested in the original research. Figure 5.1 displays the decrease in prediction accuracy as a result of the increase in attack strength. These graphs include all our obtained results for the FGSM and BIM attacks.

The most notable observation from the results presented in Table 5.1 is the significant difference between our findings and those reported in the original paper, specifically on the less complex MNIST dataset. Our reproduction results show better resilience for both models against FGSM and a significantly smaller difference between softmax and softRmax than reported in the original paper, with the softmax model even outperforming softRmax under BIM attack for MNIST.

The impact of softRmax is most significant on the more complex datasets. For the fashion dataset, softmax is outperformed by softRmax in both FGSM and BIM. With the performance gap between the two models being most pronounced for CIFAR10. While both models perform poorly on CIFAR100, the results are consistent with those reported in the original paper.

Dataset	Function	Clean (Our)	Clean (Their)	FGSM (Our)		FGSM (Their)	
				$\epsilon = 0.1$	$\epsilon = 0.3$	$\epsilon = 0.1$	$\epsilon = 0.3$
MNIST 3&7	softmax	99.80	99.75	99.26	95.49	77.18	18.69
	softRmax	**99.95**	**99.95**	**99.31**	**95.93**	**95.88**	**88.71**
MNIST	softmax	98.95	96.80	**96.48**	82.23	48.30	0.41
	softRmax	**99.14**	**97.78**	95.91	**88.41**	**75.55**	**49.30**
Fashion	softmax	**97.30**	N/A	36.71	15.54	N/A	N/A
	softRmax	94.61	N/A	**60.30**	**39.59**	N/A	N/A
CIFAR10	softmax	88.22	**80.31**	25.57	20.18	17.62	13.95
	softRmax	**88.68**	80.28	**59.92**	**47.20**	**49.93**	**41.03**
CIFAR100	softmax	**57.67**	**61.43**	11.30	7.81	11.23	6.78
	softRmax	56.70	61.04	**17.98**	**10.97**	**19.31**	**11.04**

Dataset	Method	BIM (Our)		BIM (Their)	
		$\epsilon = 0.1$	$\epsilon = 0.3$	$\epsilon = 0.1$	$\epsilon = 0.3$
MNIST 3&7	softmax	**99.26**	**95.39**	71.64	0.24
	softRmax	99.17	91.91	**94.90**	**66.24**
MNIST	softmax	**92.25**	**43.77**	53.49	0.02
	softRmax	59.32	12.21	**69.73**	**33.94**
Fashion	softmax	47.46	6.89	N/A	N/A
	softRmax	**63.53**	**28.38**	N/A	N/A
CIFAR10	softmax	6.32	0.60	10.39	4.83
	softRmax	**50.97**	**21.76**	**44.25**	**18.11**
CIFAR100	softmax	2.24	0.27	1.94	0.05
	softRmax	**8.96**	**2.72**	**9.06**	**2.16**

Table 5.1: Accuracy results (as percentage) for networks with softmax and softRmax activation under FGSM (top) and BIM (T=10) (bottom) attacks on five datasets. It shows the results of our reproduction next to the results from the original softRmax paper, listed under 'Their'.

In addition to performing BIM with 10 iterations, we also tested the CIFAR10 models with 50 iterations to assess along with the black-box attacks if the improved robustness of softRmax does not rely on gradient obfuscation. Using an epsilon value of 0.1, the experiment resulted in accuracies of 26.28% for softRmax and 0.57% for softmax. This strongly suggests that softRmax's robustness does not rely on gradient obfuscation.

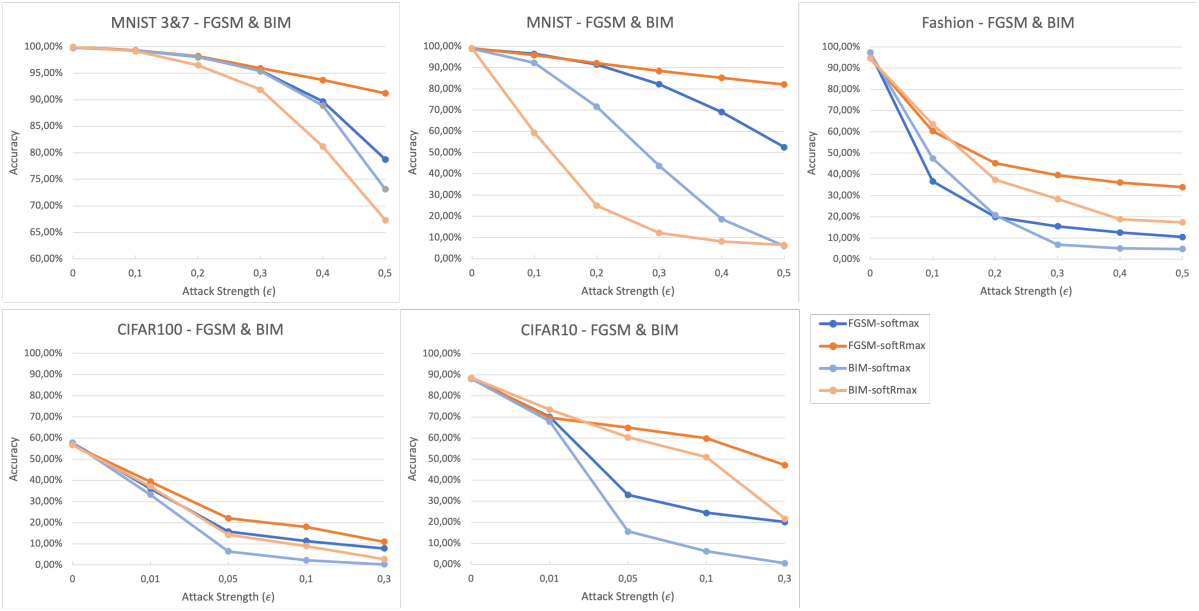


Figure 5.1: Performance comparison between softmax and softRmax activation under FGSM and BIM attack for 5 datasets. **Note:** Different y-axis scale for MNIST 3&7

In Figure 5.2, the influence of the activation function on the gradient is visualised with the final adversarial images. The figure shows an interesting distinction between FGSM and BIM. While BIM is significantly stronger (as shown in the results), it produces adversarial images that are visually closer to the originals for MNIST and Fashion. While the various patterns of noise are unlikely to affect a human's prediction regarding the class of an image, they do impact that of classification models.

In Figure 5.2, we visualised the influence of the activation function on the final adversarial images. The figure shows an interesting distinction between FGSM and BIM. While BIM is significantly stronger (as shown in the results), it produces adversarial images that are visually closer to the originals for MNIST and Fashion. Although the differences in noise patterns between the adversarial images perturbed for the softRmax and softmax models are noticeable, they are unlikely to affect a human's prediction. However, as evidenced by the results, these differences can significantly impact that of a classification model..

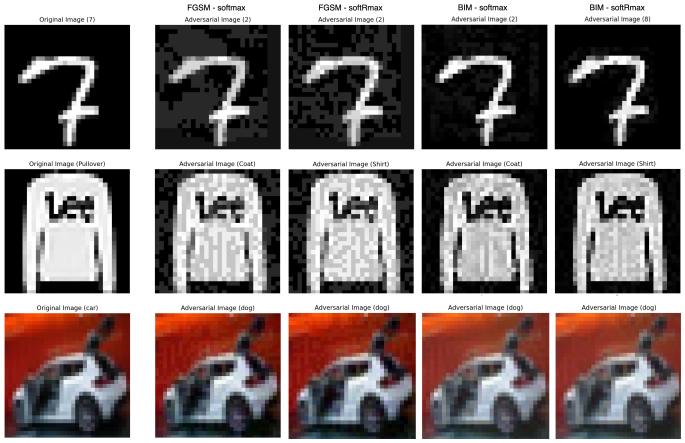


Figure 5.2: Adversarial image comparison for FGSM and BIM on the MNIST, Fashion, and CIFAR10 datasets

5.1.2 Targeted

Table 5.2 shows our results for the three targeted attacks featured in the original research. As mentioned in the previous chapter, the authors only performed these attacks on the MNIST dataset, so we can only compare those results. Additionally, it is important to note that our results are likely not directly comparable to those of the original paper, as the value of the epsilon parameter used in their targeted experiments was not specified.

The results indicate that both models perform similarly in all instances of the two black-box attacks. However, against the targeted white-box attack, the softRmax model demonstrates increased robustness across all datasets.

The results tell us that the targeted FGSM attacks are stronger than the average-sample attack in most instances. This contradicts the findings of the original paper, which emphasises the average attack as the strongest attack against the softRmax model.

5.2 New experiments

Moving on to the results of the new experiments. As explained in Section 4.3.2.1, the adversarial robustness under the two new attacks will not be assessed on the classification accuracy. Instead, we measure robustness with the mean squared error (MSE) and a function utilised in the DeepFool paper, which we refer to as the Deepfool measurement (DFM). The prediction accuracy however is still included in the DeepFool results, as that attack can have difficulties with converging some of the images (less than 2%) within the maximum amount of iterations, for our experiments the limit was set to 50. Additionally, we provide the average number of iterations it took the attacks to have the model misclassify the images.

Dataset	Function	Average Attack	Targeted FGSM	
			White	Black
MNIST 3&7 (eps = 0.5)	softmax	97.51	71.64	95.86
	softRmax	**97.95**	**94.90**	**97.60**
MNIST (Our) (eps = 0.5)	softmax	**92.73**	58.17	84.05
	softRmax	82.55	**82.52**	**85.50**
MNIST (**Their**) (eps = unk.)	softmax	24.99	10.94	33.36
	softRmax	**47.80**	**52.02**	**62.59**
Fashion (eps = 0.5)	softmax	**78.66**	11.70	**62.10**
	softRmax	74.03	**42.24**	59.40
CIFAR10 (eps = 0.3)	softmax	82.03	12.30	**44.09**
	softRmax	**82.15**	**50.57**	41.18
CIFAR100 (eps = 0.3)	softmax	47.72	12.21	13.70
	softRmax	**48.29**	**15.88**	**14.49**

Table 5.2: Accuracy results (as percentage) for networks with softmax and softRmax activation under the average-sample attack and two variants of targeted FGSM attacks on five datasets. It shows the results of our reproduction along with the results for MNIST from the original research [1], listed under 'Their'. The epsilon value used in their experiment is unknown.

5.2.1 DeepFool

The results of the DeepFool attack are listed in Table 5.3. They show that the softRmax models perform better in almost all instances. It is important to note that the percentage representing the difference between the models should not be interpreted as softRmax being 400% more robust than softmax; it serves as a more convenient way of comparing these results.

The findings indicate that MNIST requires more significant perturbations to fool the classifiers, consistent with the results of our reproduction experiments where MNIST consistently showed the lowest drop in accuracy under attack. An exception to this trend is seen in the CIFAR100 results, where softRmax exhibits the highest DFM score across all datasets. Despite the substantial difference in scores between the two metrics, both metrics show a similar relative advantage of softRmax over softmax percentage-wise.

Just like in the previous experiments, the difference between the softmax and softRmax models is the biggest for the CIFAR datasets. Interestingly, these datasets also require the fewest number of iterations, indicating that even minimal perturbations generated by the attack are sufficient to easily deceive the classifiers within a few iterations. This suggests that even smaller perturbations may be more effective against these complex datasets.

Dataset	Function	DFM	MSE	ACC	Iters (Avg)
MNIST 3&7	softmax	1.63×10^{-1}	3.13×10^{-2}	0.15%	14.02
	softRmax	$\mathbf{2.13 \times 10^{-1}}$	$\mathbf{5.73 \times 10^{-2}}$	0.23%	14.53
		31%	83%		
MNIST	softmax	1.53×10^{-1}	2.54×10^{-2}	1.22%	13.09
	softRmax	$\mathbf{1.93 \times 10^{-1}}$	$\mathbf{5.45 \times 10^{-2}}$	1.61%	13.53
		27%	144%		
Fashion	softmax	4.41×10^{-2}	3.04×10^{-3}	0.17%	7.09
	softRmax	$\mathbf{5.07 \times 10^{-2}}$	$\mathbf{3.42 \times 10^{-3}}$	0.21%	8.71
		15%	12%		
CIFAR10	softmax	1.56×10^{-2}	4.83×10^{-4}	0,00%	3.19
	softRmax	$\mathbf{7.99 \times 10^{-2}}$	$\mathbf{1.51 \times 10^{-2}}$	0,12%	3.44
		414%	$>$ 3k%		
CIFAR100	softmax	1.35×10^{-2}	9.39×10^{-4}	0.00%	4.18
	softRmax	$\mathbf{2.47 \times 10^{-1}}$	$\mathbf{2.00 \times 10^{-2}}$	0.00%	3.28
		$>$ 20k%	$>$ 20k%		

Table 5.3: Robustness results for networks with softmax and softRmax activation under the DeepFool attack on five datasets. Two metrics are used to measure robustness: DeepFool measurement (DFM) and mean squared error (MSE). Higher values indicate a more robust model. (**bold**)

5.2.2 Boundary Attack

Finally, we present the results of the Boundary Attack, which is the strongest attack in our analysis. The main observation is that the results (Table 5.4) vary significantly across datasets and metrics. The DFM and MSE metrics show conflicting results for the MNIST 3&7 and Fashion datasets. While softRmax performs worse for standard MNIST according to both metrics. Overall, these findings softRmax does not improve performance models against the Boundary Attack.

These findings align with those of the black-box attacks from the reproduction experiments, indicating that softRmax does not seem to enhance model robustness against black-box attacks and may even lead to decreased performance in certain scenarios.

Dataset	Function	DFM	MSE	Iters (Avg)
MNIST 3&7	softmax	$\mathbf{2.39 \times 10^{-1}}$	6.30×10^{-3}	4.5×10^4
	softRmax	1.03×10^{-1}	$\mathbf{1.13 \times 10^{-2}}$	5.0×10^4
		-57%	82%	
MNIST	softmax	$\mathbf{1.32 \times 10^{-1}}$	$\mathbf{1.88 \times 10^{-3}}$	4.7×10^4
	softRmax	1.04×10^{-1}	1.16×10^{-3}	5.1×10^4
		-21%	-38%	
Fashion	softmax	4.26×10^{-2}	$\mathbf{4.13 \times 10^{-4}}$	4.9×10^4
	softRmax	$\mathbf{4.61 \times 10^{-2}}$	3.97×10^{-4}	4.7×10^4
		8%	-4%	

Table 5.4: Robustness results for networks with softmax and softRmax activation under the Boundary attack on five datasets. Two robustness metrics: DeepFool measurement (DFM) and mean squared error (MSE). We interpret the results as: higher number equals a more robust model (**bold**)

As a side note, the adversarial images generated by the Boundary Attack are indeed closer to the original images compared to those produced by the DeepFool attack. Except for the DFM results of MNIST 3&7, the perturbations are smaller according to both metrics and are visibly less pronounced, as shown in the example below. This observation corresponds with the claims made by the authors of the Boundary Attack [9].

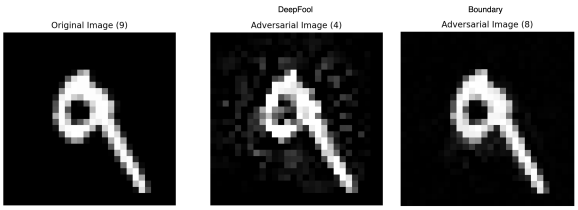


Figure 5.3: Comparison between adversarial images generated by DeepFool and the Boundary Attack for softRmax

Chapter 6

Discussion and Conclusion

In this paper, we have evaluated prior research on improving the robustness of machine learning models with softmax alternative: softRmax. We compared their influence on adversarial robustness through reproduced experiments from the original research and new experiments featuring more powerful attacks.

6.1 Findings

In the reproduction experiments, we have seen that softRmax maintained a higher prediction accuracy under attack for almost all instances of the white-box attacks. The exceptions to this came from the MNIST dataset, for which the performance under FGSM was very similar between the two models and the BIM results were significantly in favour of softmax. For the two black-box attacks, the performance between the models was comparable.

Comparing our findings to those of the original paper, the most notable difference is the significantly better performance of the softmax model for MNIST across all attacks in our experiments. For both CIFAR datasets, the models performed similarly to the results reported in the original paper.

The new experiments gave us results in the same trend as those from the reproduction experiments. The DeepFool attack demonstrated better robustness for softRmax in all instances, with the most significant performance gap being for the CIFAR datasets. In contrast, the Boundary Attack produced varied results depending on the dataset and the evaluation method, making it difficult to draw definitive conclusions

Ultimately, our findings indicate that softRmax does not offer the same level of defence against black-box attacks as it does against white-box attacks. This suggests that while softRmax's polynomial approach is effective in providing robustness against gradient-based attacks, none of our black-box experiments indicate that the claimed 'inherent gradient regularization' and resulting enlarged margin seem to enhance robustness.

Therefore, we believe that the primary factor contributing to softRmax's enhanced robustness is the function's polynomial behaviour, which provides a different structure in the tail of the distribution.

Even though our black-box attack results do not satisfy the paper's established criteria regarding gradient obfuscation, we do not see gradient obfuscation as a contributing factor to softRmax's robustness. This is supported by the results of our small high iteration BIM experiment from Section 5.1.1 and the theory presented in the original paper.

6.2 Limitations

A limiting factor in our research stems from the lack of information the original paper gives on their targeted attacks, especially the targeted FGSM variant. While we believe our recreation of the average-sample attack accurately reflects their description, its comparison is constrained by the fact that in the original research, the experiments were conducted solely on the MNIST dataset. Despite that these limitations restrict our ability to directly compare our results to theirs, they still provide valuable information on the black-box performance between softmax and softRmax.

As mentioned in Section 4.2.1.2, our experiments on CIFAR10 were not performed exactly as described in the original paper. A pre-trained VGG-16 model was used rather than a randomly initialized network. The pre-trained model has weights configured based on prior training on the ImageNet dataset, providing a strong baseline for subsequent training on other image classification datasets. We opted for this pre-trained network due to training issues with both activation functions. The consequences of this change are difficult to determine without a direct reference. However, our results did not reveal any significant outliers and showed robustness improvements similar to that observed with the Fashion dataset. This suggests that the use of the model did not adversely affect the robustness evaluation between softmax and softRmax.

6.3 Future work

Future studies could focus on extending our research with additional experiments specifically targeting black-box attacks. These experiments would provide further validation or disprove of our critique of the original research. Additionally, research could be broadened by including other softmax alternatives, such as sparsemax [3] and Taylor-softmax [4], mentioned in the introduction. Although these alternatives were not developed with robustness in mind, investigating this aspect could potentially lead to interesting new insights.

In the conclusion of the original softRmax paper, the authors suggested studying the potential of combining softRmax with other adversarial defence strategies. We agree with their suggestion and believe that techniques such as adversarial training [5] and defensive distillation [6] could potentially benefit from the conservativeness of softRmax, offering added defence against white-box attacks.

6.4 Conclusion

Using the findings presented in this study, we address the research question: *To what extent does softRmax position itself as a robust alternative to softmax for image classification tasks?*

We conclude that softRmax is a suitable replacement for softmax in scenarios where robustness against white-box attacks is crucial. However, the uncertain results regarding adversarial defence against black-box attacks prevent us from decisively recommending or opposing softRmax as an alternative to standard softmax for enhancing universal model robustness against outlying data and a variety of adversarial attacks. To be able to draw definitive conclusions, further black-box experiments are necessary.

The main takeaway from our study is that while our results contradict some of the bold claims from the original study, we, and hopefully other researchers, remain interested in the prospect of softRmax. We look forward to future research, building on our work and exploring the broader topic of adversarial robustness and (polynomial) softmax alternatives.

Bibliography

[1] Ziqi Wang and Marco Loog. Enhancing classifier conservativeness and robustness by polynomiality. *IEEE Xplore*, 2022.

[2] Ian J. Goodfellow, Jonathon Shlens and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

[3] André F. T. Martins and Ramón Fernandez Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification, 2016.

[4] Pascal Vincent, Alexandre de Brébisson, and Xavier Bouthillier. Efficient exact gradient update for training deep networks with very large sparse targets, 2015.

[5] Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. Recent advances in adversarial training for adversarial robustness, 2021.

[6] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks, 2016.

[7] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples, 2018.

[8] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks, 2016.

[9] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models, 2018.

[10] Vladimir Vovk. The fundamental nature of the log loss function, 2015.

[11] Nemo Ingendaa. softRmax analysis. `https://github.com/NemoIng/softRmax-analysis`.

[12] comp.ai.neural-nets FAQ. Should i normalize/standardize/rescale the data? `http://www.faqs.org/faqs/ai-faq/neural-nets/part2/section-16.html`.

[13] Ziqi Wang. attack. `https://github.com/ziqiwangsilvia/attack`.

[14] Pankaj. Fashion mnist with pytorch. https://www.kaggle.com/code/pankajj/fashion-mnist-with-pytorch-93-accuracy.

[15] Torchvision dataset libary. https://pytorch.org/vision/0.9/datasets.html.

[16] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[18] ewatson2. DeepFool Implementation. https://github.com/ewatson2/EEL6812_DeepFool_Project/tree/main.

[19] foolbox. Boundary Attack Implementation. https://github.com/bethgelab/foolbox/blob/bb56af1d215572d8d468c4759e8d3f5b3acfeb65/foolbox/attacks/boundary_attack.py.