UNIVERSITY OF EXETER

Computer Science Department

B.Sc. Computer Science
Computer Science Department

Siamese Neural Networks for One-shot Adversarial Robustness

Candidate

Harry Collins

Student ID 255966

Supervisor

Dr. Achim Brucker

University of Exeter

Academic Year
2022/2023

Abstract

Neural network models are a form of artificial intelligence that push the boundaries of what can be done with computers. These models have been widely adopted in everyday life in recent years, including in security-critical applications such as self-driving car systems, yet adversarial examples can be used to attack them. Adversarial examples are input that look ordinary to humans, but are confidently misclassified by neural networks meaning they make the wrong decision. There have been efforts to produce neural networks that are robust to these adversarial examples by applying defensive measures, but as of yet, there are no universally robust defenses that can defend against any kind of adversarial attack. Moreover, they require a large dataset of adversarial examples to learn from and often lack transferability to defend against attacks other than what they have specifically trained for. We propose a novel one-shot learning approach to overcome these challenges, using a siamese neural network defence that offers robustness against adversarial examples with very little training data. This defence can be incorporated into any existing image classification neural network. Once trained, the network can be further tuned on new adversarial examples and even entirely new classes of attack. It can learn to provide adversarial robustness from just a single adversarial example per output class in its training data. Secondary and tertiary results from this paper include the development of a library of small neural networks emulating security-critical applications that can be used for testing adversarial attacks and defenses, which also contains variations of popular adversarial attacks adapted in order to attack siamese neural networks. The library also contains the code necessary to reproduce our study and expand upon it.

	Yes	No
I certify that all material in this dissertation which is not my own work has been identified.	☑	☐
I give the permission to the Department of Computer Science of the University of Exeter to include this manuscript in the institutional repository, exclusively for academic purposes.	☑	☐

Contents

List of Figures iv

List of Code Snippets v

List of Acronyms vi

1 Project specification, motivation and aims 1
- 1.1 Introduction . . . 1
- 1.2 Motivation and Background Context . . . 2
  - 1.2.1 Neural Networks . . . 2
  - 1.2.2 Adversarial Attacks and Examples . . . 3
  - 1.2.3 Defences Against Adversarial Examples . . . 4
  - 1.2.4 Adversarial Machine Learning Libraries . . . 4
  - 1.2.5 Siamese Neural Networks and One-Shot Learning . . . 4
- 1.3 Project Specification and Aims . . . 5

2 Design, Methods and Implementation 6
- 2.1 Design . . . 6
  - 2.1.1 High-level Overview of Project Structure . . . 6
  - 2.1.2 Experimental Design . . . 8
- 2.2 Development . . . 11
  - 2.2.1 Developing the models . . . 11
  - 2.2.2 Developing methods to generate adversarial datasets . . . 15
  - 2.2.3 Developing variations of adversarial attacks for siamese neural networks . . . 15
  - 2.2.4 Developing the Evaluation Framework . . . 15

3 Project Results, Evaluation and Tests 15
- 3.1 Tests and Results . . . 15
  - 3.1.1 Undefended Models' Performance on Clean Datasets . . . 16
  - 3.1.2 Undefended Models' Performance on Adversarial Datasets . . . 16
  - 3.1.3 Adversarially Trained Models' Performance on Adversarial Datasets . . . 16
  - 3.1.4 Siamese Neural Networks' Performance on Adversarial Datasets . . . 16
  - 3.1.5 Mann-Whitney U Test and Cliff's delta . . . 17

4 Discussion and Conclusions 18
- 4.1 Discussion . . . 18
  - 4.1.1 Summary of Key Findings . . . 18

4.1.2 Implications 18
4.1.3 Limitations 19
4.1.4 Recommendations 19
4.2 Conclusion 20

References **20**

Acknowledgments **23**

List of Figures

1.1 These carefully manipulated stop signs, made with stickers and spray paint, can fool a self-driving car artificial intelligence system into misclassifying them as speed-limit signs[6] . . . 1
1.2 An example of a feed-forward neural network e.g. for determining whether an image is of a stop sign or not. . . . 2
1.3 A benign tumour is misclassified as malignant by a medical imaging classification neural network model after non-random, imperceptible noise is applied [13]. . . . 3
1.4 Impersonation attack: the man on the left is misclassified as the woman on the right due to his adversarial glasses[17] . . . 3
1.5 Siamese neural network example architecture[32] . . . 5

2.1 An example of a clean image (left) and an adversarial example of that image (right), generated with an epsilon of 0.2 and an $L\infty$-norm. . . . 9
2.2 The general architecture of a convolutional neural network in the style of LeNet [49]. The developed CNN models all follow a similar style to this example [50]. . . . 11
2.3 Training history for the traffic model. . . . 12
2.4 A waveform that has been converted into a spectrogram image. . . . 13

3.1 The accuracy of each undefended model on clean datasets, and their mean accuracy against adversarial datasets. . . . 16
3.2 A one-shot trained siamese verification model for MNIST's accuracy against each attack. The model was trained for MIM adversarial examples. . . . 17
3.3 A box-plot to represent the results of the Mann-Whitney U test comparing the accuracies of the traditionally adversarially trained models and the siamese verification models against adversarial datasets in a one-shot learning scenario. . . . 17

List of Code Snippets

2.1 Siamese Verification Network Architecture 14

List of Acronyms

AI Artificial Intelligence

BIM Basic Iterative Method

CNN Convolutional Neural Network

FGSM Fast Gradient Sign Method

MIM Momentum Iterative Method

PGD Projected Gradient Descent

1

Project specification, motivation and aims

1.1 Introduction

In recent years, machine learning (ML) has become increasingly adopted in many commercial areas of technology and science, such as computer vision, the development of autonomous cars, speech processing, natural language processing and robotics [1]. ML is a subfield of artificial intelligence that can be described as the study of algorithms that improve automatically with experience and without human guidance [2]. Developments in ML systems, such as neural networks that can learn to perform highly complex tasks, have led to their widespread usage due to their extremely strong performance [3]. They are trusted with tasks where lives are at stake, such as predicting whether a hospital patient has cancer [4] and driving cars. However, it has been found that a neural network's decision could be dramatically altered with tiny, practically invisible changes to the input data, known as perturbations [5]. For example, adding carefully placed stickers or graffiti to a stop-sign can cause a neural network to believe it is seeing a speed-limit sign, which could cause a huge car accident [6]. These perturbations are often imperceptible to the human eye, and are thus difficult to manually detect. Such instances are known as *adversarial examples*. Adversarial machine learning is an active

Figure 1.1: These carefully manipulated stop signs, made with stickers and spray paint, can fool a self-driving car artificial intelligence system into misclassifying them as speed-limit signs[6]

.

research area, particularly attacking and defending security-critical neural network models. Attacks and defences are evolving in a constant security arms race [7], and as such, new contributions are always required on both sides.

Many defences have been proposed to increase the robustness of neural networks against adversarial examples, but as of yet, none have been shown to be universally robust against any kind of adversarial attack [8]. Indeed, the most common flaws with defences are that they are too specific to defending against a single attack that they have been trained to defend against, therefore lacking transferability to wider application, or that they can be bypassed by increasing the strength of an attack. Most critically, they generally require a large set of adversarial examples to be used as training data [5], which may not be possible to acquire in the case of a novel attack. In such a case, there may be as few as one instance of a particular kind of adversarial example to train on, which current

defences cannot use in any meaningful way. Learning from a single instance of a class is known as one-shot classification [9].

In order to overcome these challenges, this paper proposes the use of a siamese neural network to defend other neural networks against adversarial examples. To the best of our knowledge, this is the first time that a defence of this kind has been proposed. This verification network receives the input data e.g. an image, and prediction of any neural networks performing security-critical classification tasks that would otherwise be vulnerable to adversarial attacks, and pairs that input data with an input from the predicted class. The siamese network takes these two inputs and determines whether they are of the same class. The benefits of this approach are that, once trained, the network can easily be tuned on new adversarial examples and new classes of attack, requiring as few as a single example from an attack to provide an effective defence against it, as opposed to alternatives such as adversarial training, which require thousands of examples and take a long time to retrain. This makes the siamese defence more adaptive than other defences that are difficult to scale and generalise. It can effectively learn from a single adversarial example per output class of the neural network it is defending- known as one-shot learning- providing robustness in situations where the most popular adversarial defence, traditional adversarial training [5], cannot.

Additional developments from this paper include the development of a library of small neural networks emulating security-critical classification tasks with a framework for testing adversarial attacks against them. This is useful as there are currently very few adversarial machine learning libraries containing pre-trained, small neural networks that can be readily attacked and defended. Therefore, this has been developed to streamline the testing of our proposed defence. Furthermore, included in this library are variants of popular adversarial attacks that have been developed and adapted to be able to attack siamese neural networks. This is necessary as adversarial machine learning libraries such as CleverHans [10]contain popular attacks, but the attacks are designed only to attack single-input neural networks, whereas the siamese neural network is based on pairs of inputs. To the best of our knowledge, these attacks have not been adapted to generate adversarial examples for siamese neural networks before.

1.2 Motivation and Background Context

1.2.1 Neural Networks

Artificial neural networks, commonly referred to as neural networks, are a machine learning method in artificial intelligence that are inspired by the brain. Computations are performed through the simulation of a large, layered network of connected model neurons called perceptrons, with algorithms applied to them that mimic brain functions in order to 'learn' from information to solve many complex, non-linear problems with minimal human assistance [11]. The strength of connections between these neurons, known as synaptic weights, store the knowledge learnt by the network.

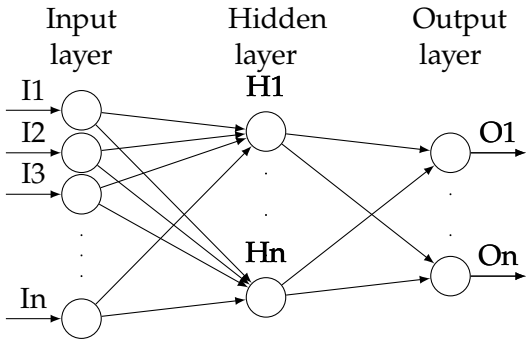

Figure 1.2: An example of a feed-forward neural network e.g. for determining whether an image is of a stop sign or not.

To perform a task, a neural network model must be trained on training data e.g. a labeled set of images, and the model's synaptic weights continually change throughout the training process so as to minimise the error/loss of the model. Once trained, these models can be applied to unseen data with very high accuracy. Therefore, they have seen widespread use, such as in photo apps, virtual assistants, and autonomous vehicles [12], with a particular kind, convolutional neural networks (CNNs) specialising in classifying images. Although highly accurate, the output of complex neural network models can be difficult to explain due to the number of self-governed computations. The resulting black-box models are often blindly trusted due to being highly accurate, but failing to understand the model's decision-making process can lead to vulnerabilites.

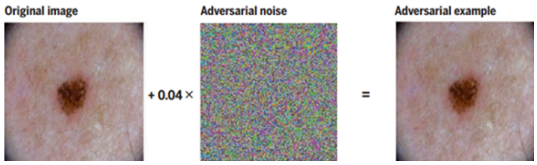

Figure 1.3: A benign tumour is misclassified as malignant by a medical imaging classification neural network model after non-random, imperceptible noise is applied [13].

1.2.2 Adversarial Attacks and Examples

Building on Szegedy et al. (2014) [14], Goodfellow et al. (2015) [5] demonstrated that neural networks can be reliably attacked using simple, computationally inexpensive algorithms which generate adversarial examples with imperceptible perturbations. Non-random noise is applied to the image, such as in Figure 1.3 which then fools the neural network into making incorrect predictions extremely confidently. These adversarial examples can be applied both digitally and in the real-world [15], thus having the potential to be highly dangerous if applied to neural networks in security-critical settings. Examples include an autonomous vehicle misclassifiying a slightly perturbed stop-sign as a speed-limit sign [16], or a criminal face-recognition database incorrectly incriminating an individual if the actual criminal is wearing a pair of adversarial glasses [17].

Figure 1.4: Impersonation attack: the man on the left is misclassified as the woman on the right due to his adversarial glasses[17]

The most common adversarial attack-type is the **Evasion Attack**, forcing a misclassification by feeding an adversarial example into a trained neural network[18]. Moreover, they can be white-box or black-box [7], and are either targeted, forcing a specific misclassification, or untargeted, simply aiming for any misclassification.

As the focus of the project is defending pre-trained neural networks, other types of attack, namely poisoning (adversarial examples are hidden in a training set) and exploratory (querying a model to discover information) attacks [18] will be ignored, and white-box attacks will be prioritised due to their higher strength.

White-box attacks typically follow a similar attack pattern to the Fast Gradient Sign Method (FGSM), one of the most popular adversarial attacks [19] due to it's simplicity and effectiveness against classification models. Developed by Goodfellow et al. (2015) [5], the attacker uses the neural network's error/loss gradient for a given input, and adjusts pixel values to maximise the loss. The expression to create such perturbations is shown:

$$\eta = \epsilon sign(\nabla x J(\theta, x, y)).$$

θ represents the parameters of a given model, x is the original input image, y is the original input label of x, $J(\theta, x, y)$ is the loss used to train the neural network, ϵ is a multiplier to ensure that perturbations are small and imperceptible to the human eye, and η is the generated perturbations. An adversarial example is then created by adding the perturbations to the original image: $adv_x = x + \eta$.

ϵ values typically fall between 0.01 and 0.1. The greater the value, the more likely it is that the perturbations will fool the model, but they will also be more visible to humans.

When testing our proposed defence, we apply the FGSM attack, as well as several more powerful white-box attacks. These will be detailed in the design, methods and implementation.

1.2.3 Defences Against Adversarial Examples

Due to the dangers of adversarial examples in real-world security-critical scenarios, neural network models must apply defences to increase their robustness against adversarial attacks. Many novel approaches have been proposed, such as Defensive Distillation [20] and Feature Squeezing [21], but although they provide a universal defence, they are not robust [22].

The most popular and well-known defence is **Adversarial Training** [5] , which involves injecting adversarial examples into a model's training set so as to learn to classify adversarial examples correctly during training. However, adversarial training is not universal, so models must be retrained for each new attack [18], which is computationally expensive both in terms of training time and in generating a large number of adversarial examples for each attack to be used in the training set. If a new attack arises that is not yet understood, it may not be possible to generate enough examples to be useful in the training data. Additionally, this defence can be bypassed in several ways, such as with transferability [23] and by combining random noise with an attack [24]. Therefore, although adversarial training provides a strong initial defence against attacks it has specifically trained to defend against, it is a computationally expensive endeavour that can be bypassed by new attacks a model has not trained for, and this approach requires a long time to retrain for new attacks, as well as a long time to gather enough adversarial examples to provide a meaningful defence.

Other emerging defences include formal verification, which seeks to formally guarantee certain behaviours in a neural network [25], although this approach is currently limited to very small neural networks.

1.2.4 Adversarial Machine Learning Libraries

To aid in adversarial machine learning research, several libraries exist which are useful for benchmarking the performance of neural networks against adversarial attacks, such as **CleverHans** [26], written in the Python programming language and compatible with popular machine learning libraries such as TensorFlow [27]. However, there are two notable absences from these libraries. Firstly, they lack dedicated, small, pre-trained models that are easy to understand and use to aid research into new attacks and defences. The libraries are also only accessible on a code-level, with little guidance for users, which could be useful in helping to understand this complex field in a similar vein to the TensorFlow Playground web-app [28] which has a user-friendly interface to interact with neural networks. There are very few intuitive tools like this, with libraries instead simply providing the code for attacks and defences. This is certainly an area that could be improved upon.

1.2.5 Siamese Neural Networks and One-Shot Learning

Siamese neural networks, first introduced to detect forged signatures [29], measure the semantic similarity between two inputs. A siamese neural network consists of a pair of neural networks with shared weights, and the similarity between the outputs of each neural network are compared to determine whether they are of the same class or not [30]. They are used to determine whether a pair of inputs are similar or not, and they excel compared to other neural networks when learning from small data sets, such as when very little training data exists, known as few-shot learning. This can occur in new scenarios where training data is not readily available, such as diagnosing patients with COVID-19 at the start of the pandemic[31], or in circumstances where it would be impractical to have

hundreds of instances of a class in a training set, such as in a facial verification system. As seen in figure 1.5, the output is a similarity score, which is then converted into a true or false label depending on what similarity threshold is set. For instance, with a 0.5 threshold, all pairs with a similarity score lower than 0.5 are predicted to be true matches.

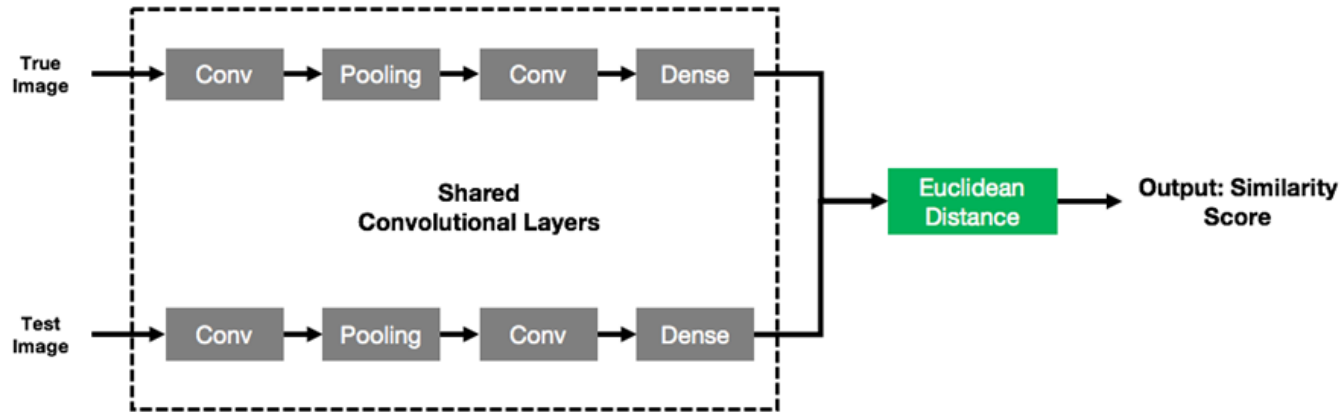

Figure 1.5: Siamese neural network example architecture[32]

.

Indeed, siamese neural networks have proven effective at image classification tasks with as few as just a single image of each class to learn from, known as one-shot learning [9]. Therefore, such an approach would be very promising if applied to classifying adversarial examples, as current approaches require large datasets of adversarial examples to learn from. These may not be possible to generate in the event of a new or novel attack that defenders cannot easily generate thousands of examples for to use in adversarial training, as the attack's code may not be freely available. In this circumstance, defences would have to be able to learn from only the adversarial examples used by the attacker, potentially resulting in a one-shot learning scenario that current defences are not effective in.

1.3 Project Specification and Aims

Main Aim

The main aim of the project is to develop a siamese neural network architecture that can defend otherwise defenceless classification neural network models against adversarial examples in a one-shot setting. It must be able to learn from small datasets of ordinary input data (1000 or fewer data points) [33], injected with a single adversarial example for each of the defenceless classification model's possible outputs. In a practical setting, after an input has been passed into a neural network and the network has made its prediction as to the input's class, the original input and a random example from the dataset of the predicted class can be passed into the siamese neural network as a pair of inputs. Then, the siamese network should correctly determine whether the inputs are indeed of the same class or not. However, to test the viability of this siamese network approach in this project, we will simply generate a dataset of adversarial examples for the defenceless model, and pass these examples into our siamese neural networks to test their robustness. The way that we test our approach will be further detailed in the Design, Methods and Implementation section.

If successful, the defence should be robust against new attacks after one-shot adversarial training, therefore offering a more adaptive, universal and less computationally expensive alternative to traditional adversarial training approaches. To demonstrate this, we will compare the performance of the neural networks that we will develop and use for testing when completely undefended, when defended by the siamese neural network, and when defended by traditional adversarial training. A single adversarial example for each output class will be injected into the training data for both defences for a fair comparison as to how capable each defence is at providing robustness through

one-shot learning.

Secondary Aims

As earlier discussed, there are a lack of adversarial machine learning libraries which demonstrate attacks on security-critical applications, and most do not contain any pre-trained neural networks to test new attacks and defences against, nor are they accessible to non-experts. Therefore, the first of the secondary aims of the project is to develop a library of small neural networks emulating security-critical classification tasks: identifying road signs, voice commands, hand-written digits and facial recognition/verification. The neural networks will be relatively small in terms of the number of neurons and layers so as to be understandable and quick to train and retrain. Therefore, they will be useful for testing new attacks and defences due to their simplicity and the adversarial examples produced will emulate potential real-world applications of adversarial examples. Moreover, their small nature also allows them to be useful in research into formal verification [25] of neural networks, which is not the case for other adversarial machine learning libraries which focus on standard attacks and defences.

Moreover, to make the library more accessible and understandable than other libraries, we will use Jupyter Notebooks [34] to write our code, allowing users to easily follow along with the GUI-like structure of such documents whilst also maintaining the complexity required to develop the library.

Another aim is to adapt existing adversarial attacks to be able to attack siamese neural networks. Current attacks, such as those curated in the popular adversarial machine learning library CleverHans, attack a single input, and thus do not work to their full potential against siamese neural networks, which measure the similarity between two inputs. When considering the emergence of siamese neural networks for one-shot learning tasks, it is necessary to create variants of existing attacks that create adversarial examples for input pairs to properly test the capabilites of such neural networks, including our proposed defence. As a result, we will adapt the adversarial attacks for TensorFlow in CleverHans that we use in our study, and create variants to generate adversarial examples for input pairs to be used against siamese neural networks.

To combine all of these goals, we will combine the pre-trained models, siamese attack variants and our siamese verification networks into a single library, which we will name the 'LittleAdversary' library to reflect the small nature of the neural networks and the adversarial examples that will be produced for them. We will go into specific detail on the library's structure in the design section.

2 Design, Methods and Implementation

2.1 Design

2.1.1 High-level Overview of Project Structure

The project will be structured as a library with four modules: models, data, graphs and notebooks.

Models will contain the siamese adversarial verification neural networks as well as four pretrained small neural network models for the following classification tasks: handwritten digit recognition, traffic sign recognition, facial recognition and speech command recognition. These models will also have corresponding adversarially trained model variants. The code to build these models will be contained in Jupyter notebooks in the 'notebooks' module that will contain a structured pipeline to build these neural networks, allowing an end-user to make adjustments or create their own neural networks based on those in the library. Each model will have two corresponding folders: one containing the traditionally adversarially trained model weights for the model, and another for the siamese verification weights for that model.

To be considered successful, the four pretrained classification models must attain an accuracy of over 85% on test data from their corresponding datasets. The siamese neural network must, after undergoing one-shot adversarial training, be able to correctly classify adversarial examples with far greater accuracy than a standard CNN that has undergone one-shot adversarial training. We define one-shot adversarial training as adversarial training that is limited to only injecting a single adversarial example for each ouput class into the dataset, replicating a scenario where there is no easy way to create a large dataset for a certain kind of adversarial example, such as for a new attack.

Data will contain the datasets required to train each of the models. The datasets used are as follows:

- MNIST handwritten digits dataset [35]. This is a commonly used dataset for developing image classification neural networks, featuring 70,000 images of the handwritten digits 0-9, resulting in 10 classes. Each image is 28x28x1. It has been chosen due to its popularity and for the fact that it replicates the potentially security-critical application of reading and classifying handwriting.
- The German Traffic Sign Recognition Benchmark dataset [36]. This contains over 50,000 30x30x3 images of 43 classes of traffic signs taken in Germany. It has been selected due to the variety of potential inputs and the fact that the data can be applied to the security-critical task of traffic sign recognition for autonomous vehicles.
- The Speech Commands Dataset [37]. We use a simplified version of the original dataset, containing 8000 .WAV files for 8 classes of speech command e.g. yes, no. The main appeal of this dataset is classifying inputs beyond images, thus adding variety to the library and to the defensive capabilities of the siamese neural network.
- The AT&T Database of Faces [38]. This is a small database containing 400 92x112x1 images, 10 for each of the 40 different faces. With 40 classes and so few images, this dataset presents a different challenge to the other, larger datasets. It was also chosen as it can replicate the security-critical task of facial verification.

As can be seen, each dataset enables the corresponding neural network model to perform a security-critical task, and the variety in terms of dataset size, data types, and number of classes means each model will be distinct from the other, thereby increasing the viability of using the library for testing adversarial attacks and defences.

This module will also contain adversarial datasets in the subfolder 'adv_datasets', which will be generated in our adversarial evaluation notebook.

Notebooks will contain the Jupyter notebooks that show how to build each neural network model, which will be titled 'Building a ... NN'. Moreover, the code we use to develop our traditionally adversarially trained models and our siamese verification networks will be contained in a notebook entitled 'Adversarial One-shot training'. These notebooks will also show how to access, save and load the pre-trained models. We will also include a notebook containing our attack variants, called 'siamese_attack_variants'.

Siamese_attack_variants will contain variants of attacks imported from the CleverHans library that we will develop to specifically attack siamese neural networks. These attacks will be used to test the performance of the models against adversarial examples, and to test the performance of the siamese adversarial verification networks against traditional adversarial training.

The largest notebook, 'Adversarial Defence Experiment, Evaluation and Testing' will contain the end-to-end tests for testing the performance of the models, including their performance against a

clean dataset, performance against each attack, and performance against attacks when defences are applied. This will provide the data to determine whether the proposed defence is effective or not, and how it compares to traditional adversarial training through structured statistical testing.

The notebooks module will also contain several utility notebooks, titled as 'utils...' that will be used to help the main notebooks to run, and to help users to create their own siamese verification networks and small security-critical neural networks whilst abstracting the low-level complexity of the process.

Finally, the graphs module will contain training histories for our siamese models as images, as well as various visualisations of our data and testing results.

2.1.2 Experimental Design

As adversarial machine learning is still a relatively new field of research, strict standards for testing and evaluating new attacks and defences do not yet exist, although Carlini and Wagner developed a set of guidelines to follow in order to determine the true robustness of a neural network against adversarial examples [39] which was later expanded [40]. Whilst many of these guidelines are useful, the authors note that they should not necessarily be strictly followed, and that the exact methodology for testing defences will differ for each project. As a result, we will follow the most appropriate guidelines from the paper, and adapt the general approach to our siamese neural network. Firstly, we will define our hypothesis and what tests we will perform to compare our siamese neural network approach and the traditional adversarial training approach in a one-shot setting.

Overall Test and Hypothesis

We will compare our approach to traditional adversarial training via a Mann-Whitney U test[41], a test to compare differences between two independent groups where the data is not normally distributed and where the variances between the two groups are not equal. We will compare the classification accuracies of the models that have been traditionally adversarially trained and the accuracies of the siamese verification networks against adversarial datasets. We anticipate neither a normal distribution or equal variance between groups, as we expect fairly uniform accuracies in each group, and the variance cannot be known before finding results. Both the traditionally adversarially trained models and the siamese verification models will only have their training data injected with a single adversarial example per output class. We opt for accuracy (no. of correct predictions/ all predictions) as our metric as it is easy to understand and the paired datasets we will generate for our siamese neural networks will be balanced, so accuracy will fairly represent the performance of the neural network models.

We hypothesise that the accuracies of the siamese verification networks will be significantly greater than the traditionally adversarially trained models, demonstrating that our approach can provide adversarial robustness with as little as a single adversarial example per output class in a classification problem. Our chosen significance level is the standard 0.05 [42]. Therefore, if the p-value returned by the test is less than 0.05, we will be able to conclude that there is a significant difference in performance between traditional adversarial training and our proposed approach when in a one-shot setting.

To further investigate the magnitude of difference between our proposed siamese method and traditional adversarial training in one-shot settings, we will also measure the effect size between the two approaches via Cliff's delta [43], which ranges from -1 to 1, with larger absolute values indicating a greater magnitude of difference between the approaches. We hypothesise that we will attain an absolute Cliff's delta approaching 1, meaning our approach is not only better, but improves upon traditional adversarial training by a large amount in one-shot settings. Now that the end-goal of our experiment and the tests have been outlined, we will address Carlini et al's [40] guidelines, beginning by establishing a threat model.

Threat Model

Our threat model dictates the conditions that we will design our defence to be secure in, and ensures scientific repeatability and reproducibility [40]. As such, we will define our adversary's goals, knowledge and capabilities as follows:

- Goals: the goal of our adversary is simply to produce an adversarial example that causes a misclassification, as any kind of adversarial misclassification in a security-critical setting could be disastrous.
- Knowledge: in accordance with the aforementioned guidelines, we follow the best practice of assuming our adversary has white-box access to our models, meaning complete knowledge of the models and their parameters. Therefore, attacks will be as strong as possible within the capabilities of the adversary.
- Capabilities: we restrict our adversary to making small changes to valid input data e.g. from the provided test data data for each dataset, which are imperceptible to humans, as we are building a defence against such adversarial examples. We formally define an adversarial example in this case as: where x is ordinary input and D is a similarity metric, if $D(x, x') \leq \epsilon$, x' is a successful and valid adversarial example if it is misclassified by the neural network model. Our similarity metric D is the $L\infty$-norm, which, when combined with an epsilon of 0.2, means that an adversary cannot alter any single pixel value by more than 0.2. 0.2 has been chosen as the largest epsilon value as perturbations become more obvious and less imperceptible as this value increases, with more of an appearance of noisy data rather than an adversarial example. This large norm-bound should prove challenging for a defence to overcome. Another norm that could be explored in future research is the $L2$-norm, which essentially results in fewer modified pixels, but larger perturbations on those few.

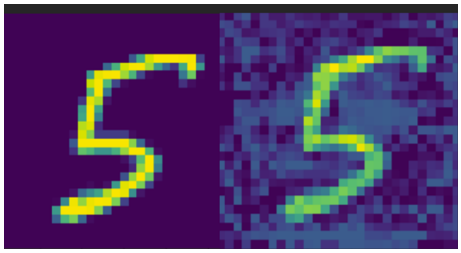

Figure 2.1: An example of a clean image (left) and an adversarial example of that image (right), generated with an epsilon of 0.2 and an $L\infty$-norm.

Additionally, it is important to determine how robustness will be measured. A common definition for robustness is the worst-case loss for a given perturbation budget [40]. We define our perturbation budget as $L\infty$-norm with ϵ of 0.2, showing an adversarial example of this kind in figure 2.1. We will be measuring the worst-case loss of the models and the accuracy of each model's predictions.

The two most important suggestions from Carlini and Wagner's original paper on evaluating adversarial robustness [39] are to test the proposed defence against strong attacks, and to demonstrate that transferability attacks will fail. To meet these most critical evaluation requirements, we will test our siamese neural network defence and the traditional adversarial training defence using the white-box attacks in the CleverHans library's TensorFlow attacks module that use the $L\infty$-norm, as they are both powerful due to their white-box nature, and fall under our threat model. The attacks are as follows:

- Fast Gradient Sign Method (FGSM): as described earlier, this attack is popular and produces adversarial examples very quickly. Although a weaker attack, it will serve as an interesting comparison with the more powerful attacks listed below.
- Basic Iterative Method (BIM) [15]: this extends FGSM, iteratively applying it to an image iteratively multiple times. Whilst slower, the resulting perturbations are more subtle.
- Projected Gradient Descent (PGD) [44]: this attack is similar to BIM, but begins by initialising random perturbations on the original input before iteratively tuning them.

- Momentum Iterative Method (MIM) [45]: this expands upon gradient-based iterative attacks like PGD and BIM by applying the concept of momentum, memorising and utilising the gradients from previous iterations in the attack to increase efficiency. This attack won Google Brain's NIPS 2017 Adversarial Attacks competition, demonstrating its powerful nature.

To ensure that transferability attacks will fail, we will test our defended models against adversarial datasets containing adversarial examples generated from attacks that the model has not trained for.

Experimental Structure

Listed below is the detailed structure of the experiment, informed by Carlini et al's [40] expanded guidelines for evaluating adversarial robustness.

1. **Undefended models' performance on clean datasets:** Measure accuracy and loss of undefended models on original data: Baseline figures for the performance of the undefended networks when predicting the classes of ordinary inputs will be attained.

2. **Undefended models' performance on adversarial datasets:** Measure accuracy and loss of undefended models on adversarial data: For each model and attack, generate a dataset of adversarial examples based on the original testing data for the model, and use this as input for the model to classify. The accuracy for each of the models against all adversarial datasets will be collected. This step will also confirm whether our siamese attack variants can effectively produce adversarial examples for siamese models, as we will gather the accuracy of the attacks against the siamese face classification model.

3. **Adversarially trained models' performance on adversarial datasets:** Create copies of the undefended models, and retrain them via one-shot adversarial training for each attack by injecting a single adversarial example for each class into the training set, thus training them on one-shot adversarial datasets. The cross-validated accuracy of each defended model will then be observed by calculating the accuracy of each model against each attack.

4. **Siamese neural networks' performance on adversarial datasets:** Train siamese neural networks for each dataset used by the undefended models, taking a sample of just 400 data points from the original datasets (ensuring that every output class is represented) and converting them into input pairs with boolean labels for true matches and false matches, resulting in 20000 data points. This process will be the same as will be done to train the face classification model, so will be further detailed in the development section. Then, repeat this process for the one-shot adversarial datasets to create paired adversarial datasets and retrain a new siamese neural network model for each of these datasets (the face classification model will be excluded from this retraining, as it cannot for our aim to compare standard CNN adversarial training and siamese adversarial training). Once trained, test each model against adversarial datasets for each attack, attaining cross-validated accuracy scores for the MNIST, traffic and speech models by calculating the accuracy of each of their defended models against each adversarial dataset. This also measures transferability, and we suggest that a variance in accuracy of less than 5% for each defended model against all of its adversarial datasets demonstrates that transferability attacks are ineffective, as the models will be tested against attacks they have not trained for.

5. **Mann-Whitney U Test:** As detailed in the high-level overview, we will compare the accuracy scores acquired in the previous steps for the MNIST, traffic and speech models to determine whether there is a significant difference between the accuracy of classification models defended via traditional adversarial training and models defended via our siamese approach to adversarial training in a one-shot setting. We hypothesise that our approach will significantly outperform traditional adversarial training, defining our significance level as the standard 0.05 [42]. Before running the test, we will ensure that our data does not have a normal distribution and that the variance between the groups is not equal by running the Shapiro-Wilk test [46] and Levene's test [47] respectively.

After the data has been gathered, we will generate tables and plots to analyse, visualise and summarise our findings. These will be detailed in the development section. We hypothesise that, across all tests, the siamese adversarial verification network will significantly outperform traditional adversarial training on the one-shot (single adversarial example for each class) training data.

As the main aim is to compare the performance of our proposed defence against traditional adversarial training, hyperparameters for the attacks, besides adjusting the perturbation budget as detailed, will be fixed throughout the tests. These hyperparameters are the change in perturbations in each attack iteration, and the number of iterations in the iterative attacks. These will be set to 0.05 and 4 respectively, meaning that perturbation changes are small in each iteration, and that there are enough iterations to create an ideal adversarial example.

2.2 Development

Development and implementation of the library was conducted in a modular fashion in the following order:

1. The four neural network models emulating security-critical classification tasks were developed.
2. The siamese neural network architecture to defend against adversarial examples was developed.
3. Methods were written to use the attacks from CleverHans to generate datasets of adversarial examples for the models.
4. Variants of the attacks were developed so that the siamese neural network could be attacked.

Python [48] was chosen as the programming language for writing the library so that TensorFlow, the popular machine learning library [27], could be used to develop the neural network models. In particular, we opted to use TensorFlow's Sequential API and Keras to build the models, as these high-level tools make the code easily readable and understandable.

2.2.1 Developing the models

Background research determined that the most common type of neural network models for performing classification tasks with images are convolutional neural networks. These neural networks contain convolutional layers that find patterns in images, making them suitable for this task. All of the following models were developed with time-performance in mind, so the goal was to ensure that they were reasonably small in terms of the total number of neurons and layers while still retaining high accuracy. For each model, development typically required two distinct steps: preprocessing the data, and developing and training the model. Each model and the inspiration behind the final architecture, as well as reasoning behind why certain choices were made will be detailed.

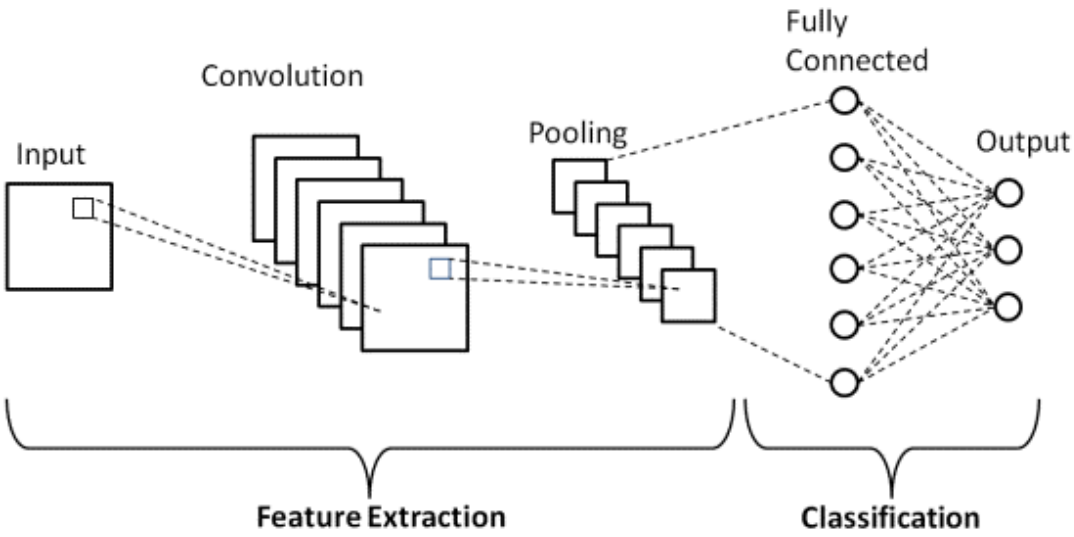

Figure 2.2: The general architecture of a convolutional neural network in the style of LeNet [49]. The developed CNN models all follow a similar style to this example [50].

MNIST Classification Model

The MNIST model takes a 28x28 input image from the MNIST handwritten digits dataset, and classifies it as a digit from 0-9.

Preprocessing: As the MNIST dataset can be imported from within TensorFlow, very minimal preprocessing was required. The dataset was also pre-split into a test set and a training set. Therefore, the only step required after importing was to normalise the image data, done by dividing the data by 255 (range of possible pixel values).

Developing and training the model: As stated, the model takes a 28x28 input , and the input layer flattens this into 784 neurons. This is followed by two fully connected hidden layers, each with 128 neurons. Both use a ReLU activation function as recommended for CNNs, although this task is simple enough that no convolutional layers are required. The final layer is the output layer, with 10 neurons, one for each possible output. The model architecture is inspired by TensorFlow's example MNIST classifier, but an additional hidden layer was added to increase accuracy, which is only ~70% on TensorFlow's model. Furthermore, L2 regularisation was added to the hidden layers to help reduce overfitting, as initial efforts to create the model performed far worse on test data than on training data. The model's loss function is sparse categorical crossentropy, which is recommended when there are two or more output classes labeled as integers. We also use the Adam [51] optimizer due to its efficiency and the fact that it is well-regarded as one of the best optimizers for deep learning. After developing the model, we trained it for 15 epochs, resulting in a model with a loss of ~0.01 and an accuracy of ~99.8% accuracy on the training data. In reality, parameters were tweaked and training was redone several times to find an optimal model that attained a high accuracy without adding too much complexity. The final model has a loss of ~0.13 and an accuracy of ~97.5% accuracy on the test data, which satisfies the requirements set out in the design section. The model was saved as 'FF_digit_classification_model_l2_regularised.model' in the models module of the library, with adversarial training weights being saved into the 'mnist_weights' subfolder of models.

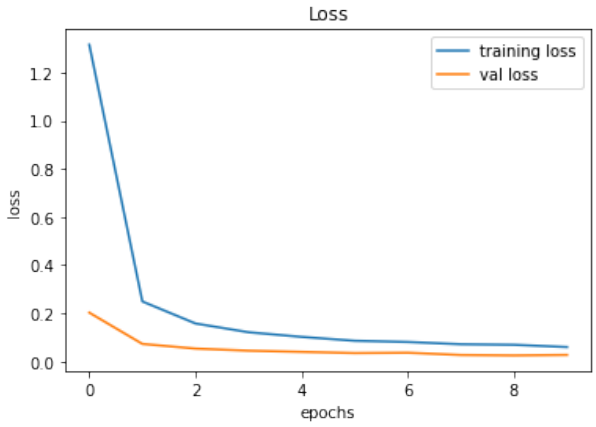

Figure 2.3: Training history for the traffic model.

Traffic Sign Classification Model

This model classifies 43 classes or traffic sign, and as a result, contains more layers and neurons than the MNIST model. For this model, we take inspiration from the LeNet CNN architecture [49], similar to figure 2.2,which has been shown to be both simple and effective. The input layer accepts a 30x30x3 input, totalling 2700 neurons, which are then passed through two convolutional layers. Each has 32 filters/neurons. 32 filters was deemed an acceptable size as the images are small, and this is the smallest common number of filters seen in practice. We also follow the common practice of moving from a large kernel size relative to our input, to a small kernel size, so for these layers we use a 5x5 kernel for our filters. Then, after a dropout layer to reduce overfitting, we repeat these layers, but with 64 filters, as recommended practice is to increase the number of filters throughout the network, and we also follow guidance to reduce kernel size, so we set ours to 3x3 for these convolutional

layers. This means that our neural network will, after learning a small number of broad features of inputs, learn a higher number of finer details to help classify the image. Then, we add a densely connected 256-neuron hidden layer in accordance with LeNet, before our 43-neuron output layer. The training history can be seen in figure 2.3. The model was saved as 'trafficsignNN_normalised.h5' in the traffic_model subfolder of the models module, and adversarial training weights for this model were saved in the 'traffic_weights' subfolder of models.

Speech Command Classification Model

This model is based on Tensorflow's 'Simple Audio Recognition' and Microsoft's 'Introduction to Audio Classification with Tensorflow' [52]. The goal is for the model to understand and recognise the following audio commands: 'stop, go, yes, no, up, down, left, right' when receiving .wav file input.

Preprocessing: The dataset contains 8000 .wav files, with 1000 for each class. Firstly, the order of the files was shuffled, before splitting the shuffled data into a train, test and validation split. 80% of the data, 6400 files, were used as training data, with test and validation being allocated the remaining 20% evenly. Then, in order to adopt a similar CNN architecture as the other models, we generated spectrogram images of the wav files, and used these images as the input for our neural network. The conversion process can be observed in figure 2.4.

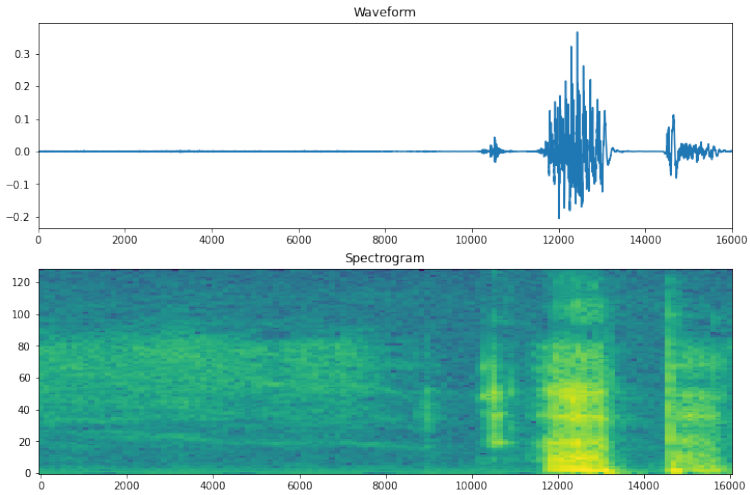

Figure 2.4: A waveform that has been converted into a spectrogram image.

Developing and training the model: The model's input layer accepts a 124x129 .wav file as input, totalling 16000 neurons when flattened. This input is then resized into a more manageable 1024 neurons. We then adopt a LeNet-style architecture once again, but, as there are fewer possible outputs and because we have reduced the number of input neurons, we simplify this architecture more than in the traffic model. After mapping the .wav file into a 32x32 spectogram image, we apply two convolutional layers to learn features, going from 32 to 64 filters and maintaining a small 3x3 kernel size, before moving to a MaxPooling layer. This layer retains the most prominent features found by the earlier layers and discards the rest, increasing efficiency and reducing overfitting. To further reduce overfitting, we include a dropout layer, before a final densely connected output layer with 8 neurons for the 8 outputs. This model had 86% accuracy on test data, just above our acceptable threshold, and is saved as 'speechcommandsNN' in models. We saved adversarial training weights for this model in the 'speech_weights' subfolder of models.

Face Classification Model

In order to create a face classification model that is not overly complex, we utilise a few-shot learning approach inspired by Koch et al's siamese neural network for one-shot image recognition [9] and Ravichandiran's face classification siamese neural network architecture [53]. The goal is to determine whether two faces are similar or not when given an input of two images of faces.

Preprocessing: The main challenge for this section was to create a method for generating a paired dataset from the data. To do this, we cycled through every possible image combination and assign a boolean label to the pairs, either 1 for a genuine match, or 0 for a false/imposite match. This means the dataset went from 400 images to 20000 image pairs, and the images were also resized to from 112x92 to 56x46 to reduce the number of neurons required in the neural network. We also took care to create a balanced dataset, with 10000 genuine pairs and 10000 imposite pairs. After this, we split the data into training and test/validation sets at a 75:25 ratio.

Developing and training the model: The siamese network contains two identical CNNs for feature extraction. Each of the two images is processed by one of the CNNs, which extract feature vectors from the images. The CNNs have an input layer that takes a 56x46x1 input image, which is then passed through a convolutional layer with 6 filters and a 3x3 kernel size as Ravichandiran recommends. We then apply a MaxPooling layer to reduce overfitting and increase efficiency, and then repeat these two layers, but with 12 filters rather than 6, for the same reasons as our other CNNs. The input is then flattened and passed into a fully connected dense layer of 128 neurons. Finally, we connect the CNNs via a TensorFlow Lambda layer to compute the euclidean distance between the two sets of feature vectors acquired from the CNNs. We expect outputs lower than 0.4 to be genuine matches, whereas outputs of a distance greater than 0.4 are not, otherwise known as imposite matches. Once trained, This model had a loss of ~0.05 and ~93% accuracy on test data. We saved the model weights as 'siamesefacepredictorweights.h5' in the face_siamese_weights subfolder of models.

Siamese Adversarial Verification Network

For our adversarial example verification network, we take the same inspirations from Koch [9] and Ravichandiran [53] to create a siamese neural network to defend against adversarial examples by computing the euclidean distance between pairs of inputs. This euclidean distance between inputs is measured by a contrastive loss function. Therefore, the architecture is similar to the face classification model, with added dropout layers after the convolutional layers and an additional dense layer. These changes were made to ensure that this model can be applied generally to a wide variety of different classification tasks without having to change the architecture each time to attain effective results. Once the model's architecture was complete, we then trained a model on each of the other neural networks' clean datasets. The general architecture's code is shown below. We build two feature extractor CNNs using the build_siamese_model method, and then use the get_siamese _model method to combine the feature extractors and add them to the overall siamese neural network. After experimenting with different threshold values, we decided to set the threshold for these models at 0.4, meaning that if the network outputs a similarity score less than this for a given input pair, it predicts a true match. We developed these models in the 'Adversarial One-shot training' notebook, along with traditionally adversarially trained models. We saved the weights in subfolders of the models module, and show the structure with an example: mnist_siamese_weights. Each subfolder starts with the name of the original model.

```
def build_siamese_model(input_shape, embedding_dim=128, conv_size=32,kernel_size=3):
    # specify the inputs for the feature extractor network
    inputs = Input(input_shape)
    # define the first set of CONV => RELU => POOL => DROPOUT
    x = Conv2D(conv_size, (kernel_size, kernel_size), padding="same", activation="relu")(
    inputs)
    x = MaxPooling2D(pool_size=(kernel_size, kernel_size))(x)
    x = Dropout(0.3)(x)
    # second set of CONV => RELU => POOL => DROPOUT layers
    x = Conv2D(conv_size*2, (kernel_size, kernel_size), padding="same", activation="relu"
    )(x)
    x = MaxPooling2D(pool_size=2)(x)
    x = Dropout(0.3)(x)
    x = Dense(128)(x)
    # prepare the final outputs
    pooledOutput = GlobalAveragePooling2D()(x)
    outputs = Dense(embedding_dim)(pooledOutput)
```

```
    # build the model
    model = Model(inputs, outputs)
    return model
```

Code 2.1: Siamese Verification Network Architecture

2.2.2 Developing methods to generate adversarial datasets

To generate adversarial datasets, we imported the selected attacks from CleverHans, and created methods that take the data, generate datasets of adversarial examples from that data for each attack, and then return those datasets.

2.2.3 Developing variations of adversarial attacks for siamese neural networks

As has been stated previously, one of the main challenges of this project was creating adversarial examples for siamese neural networks, which was needed to allow for a true white-box attack against our defence. Therefore, for each of the aforementioned attacks, we altered the CleverHans source code so that the attacks would work on pairs of inputs. We did this in two ways: firstly, applying perturbations to both images, and secondly, applying perturbations to a single image, which is an experimental feature. In our library, we include this as a hyperparameter option when running an attack, but for our experiment, we focused on the multi-image perturbations to avoid overcomplicating the testing stage of the project.

2.2.4 Developing the Evaluation Framework

To evaluate the models, we determined that a single Jupyter Notebook containing our evaluation code would be appropriate, fulfilling another of Carlini et al's evaluation guidelines [40], with this being that the evaluation should be able to be run end-to-end with a single button press. Therefore, we developed a Notebook containing all the necessary code to perform our evaluation as outlined earlier. We also included code to generate plots to visualise our results.

3

Project Results, Evaluation and Tests

3.1 Tests and Results

To test our library and our proposed defence, we followed the steps listed in our experimental design, with each step coded chronologically in a Jupyter notebook to make repeating and reproducing the test as easy as possible. Firstly, we imported our models and datasets, and then wrote helper functions to perform tests as efficiently as possible, before commencing testing and evaluation in our 'Adversarial Defence Experiment, Evaluation and Results' notebook.

3.1.1 Undefended Models' Performance on Clean Datasets

We confirmed that our security-critical neural network models worked on expected inputs by measuring the accuracy and loss of each model on the appropriate test data. The results are shown in figure 3.1. As can be seen, the models meet the success thresholds set in our design section, indicating that these small models are effective on normal input data.

3.1.2 Undefended Models' Performance on Adversarial Datasets

We also observed the undefended models' performance against datasets filled with adversarial examples to ensure that attacks work as expected, and to test that our attacks for siamese neural networks work. We cross-validated our findings by testing each model against several datasets, one for each attack. We then took the mean accuracy of the model against all datasets. As we can see in figure 3.1, accuracy drops significantly for our models compared to the accuracy on clean data.

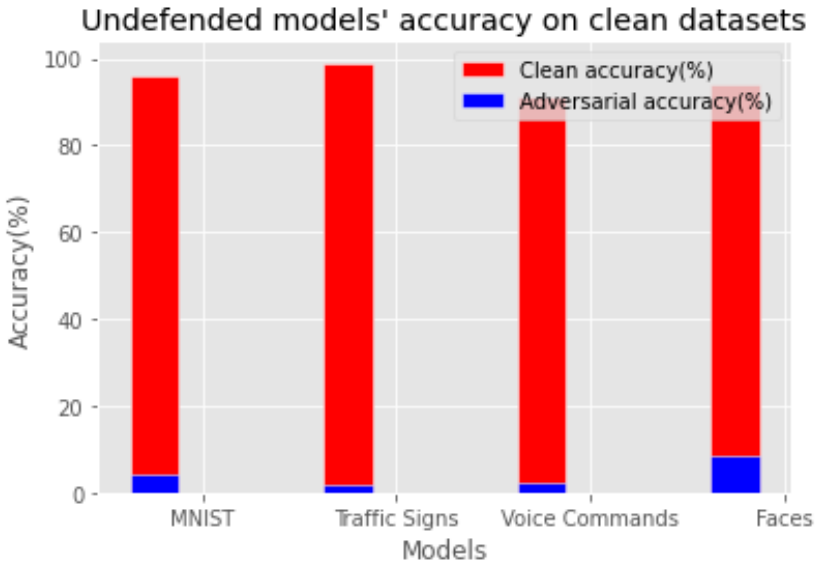

Figure 3.1: The accuracy of each undefended model on clean datasets, and their mean accuracy against adversarial datasets.

3.1.3 Adversarially Trained Models' Performance on Adversarial Datasets

Each of the standard undefended CNN models were retrained for each adversarial attack, using a single adversarial example from the given attack for each possible output class. Therefore, each model had four retrained alternatives, one for each attack. We then observed the cross-validated accuracy of the adversarially trained models against adversarial datasets from the previous step, resulting in three accuracy results: one for MNIST, Traffic and Speech Commands. These models' performance versus the siamese neural networks' performance can be seen in figure 3.3.

3.1.4 Siamese Neural Networks' Performance on Adversarial Datasets

As detailed in the experimental structure, we trained a siamese neural network counterpart for each of our initial undefended neural networks on clean datasets, and then retrained each in a similar manner as the retraining in the previous step. We then attained cross-validated accuracy scores by testing against adversarial datasets for each attack, which also measures the transferability of the defences by testing them against attacks they have not specifically trained for. We found that accuracy deviated by less than 10% when testing each model on the adversarial datasets, confirming a certain level of transferability. We demonstrate this by showing the accuracy of the siamese verification

network for MNIST, one-shot trained on MIM adversarial examples. As figure 3.2 shows, despite only being trained for one attack, the model is able to maintain similar accuracy against other attacks.

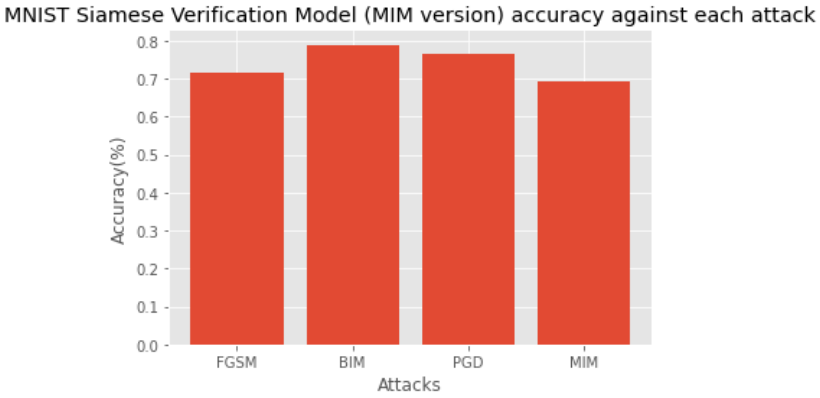

Figure 3.2: A one-shot trained siamese verification model for MNIST's accuracy against each attack. The model was trained for MIM adversarial examples.

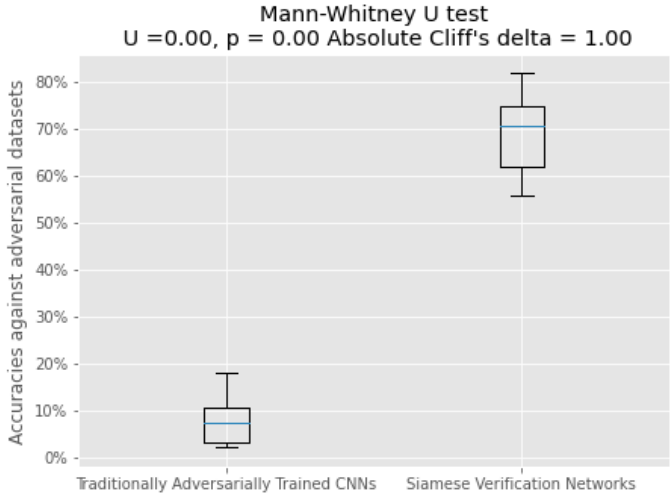

Figure 3.3: A box-plot to represent the results of the Mann-Whitney U test comparing the accuracies of the traditionally adversarially trained models and the siamese verification models against adversarial datasets in a one-shot learning scenario.

3.1.5 Mann-Whitney U Test and Cliff's delta

To compare traditional adversarial training against our siamese adversarial training approach in a one-shot setting, we performed a Mann-Whitney U test, comparing each of the accuracy scores attained for the traditionally adversarially trained models with the accuracy scores attained by our siamese verification networks. We visualise the results of the test in figure 3.3. To ensure this data did not follow a normal distribution and that the variances between the two groups of scores were not equal, we performed a Shapiro-Wilk test and Levene's test. The resultant p-values (to four decimal places) were 0.0022 and 0.0004 for the Shapiro-Wilk test, and 0.0054 for Levene's test, all falling below the standard significance value of 0.05, thus meaning our data was as expected. Once this was known, the Mann-Whitney U test was performed, returning a U value of 0 and a p-value of 0, demonstrating that a significant result was found. To find the magnitude of the difference between our approach and traditional adversarial training, we calculated the effect size via Cliff's delta, acquiring an absolute value of 1, suggesting that there is a very large difference in effectiveness between the approaches in

the siamese verification networks' favour. This significant difference is shown in figure 3.3.

4 Discussion and Conclusions

4.1 Discussion

4.1.1 Summary of Key Findings

This study's aim was to propose a novel defence for neural network models that would otherwise be vulnerable to adversarial examples. In particular, the aim was to provide a defence that is effective at increasing robustness in a one-shot setting, where only a single adversarial example exists per output class to be trained on, and we hypothesised that our defence would be far more effective in this scenario than the most popular defence at this time, traditional adversarial training.

Our results clearly demonstrate that our main hypothesis was correct, with the Mann-Whitney U test showing that there was a significant difference between the accuracies of our siamese verification networks and the traditionally adversarially trained models, with our siamese models averaging ~70% accuracy when classifying adversarial examples compared to ~7% accuracy for the traditionally adversarially trained models. The U value of 0 attained from this test also shows that every siamese model variations' accuracies were greater than their traditional counterparts. Furthermore, our Cliff's delta test confirmed, with an absolute value of 1, that there is a large magnitude between the effectiveness of our approach and the effectiveness of traditional adversarial training when it comes to one-shot learning, and such strong results from both tests suggest that this is not down to random chance, but should be consistent for any repeat tests. The significant difference between our approach and traditional adversarial training can be seen in figure 3.3. Other aims were also fulfilled, such as demonstrating that the library of pre-trained, undefended models can effectively replicate security-critical classification tasks as they all performed at over 85% accuracy, visualised in figure 3.1, and we also completed our goal to create attack variations to specifically attack siamese models, shown as we successfully generated adversarial examples for the face classification model. This can also be observed in figure 3.1. Moreover, figure 3.2 shows that our siamese verification networks have reasonable levels of transferability, as accuracy against adversarial attacks that the siamese models had not trained for was similar to accuracy against the attack each model specifically trained for.

4.1.2 Implications

The results indicate that our siamese verification network defence provides an effective defence in a one-shot setting where training data is extremely limited, as the models were able to learn to correctly classify adversarial examples with ~70% accuracy despite only learning from a sample of 400 data points, with only a single adversarial example per output class injected into that sample. On the other hand, results show that traditional adversarial machine learning was ill-suited to this task, failing to learn to correctly classify adversarial examples. Therefore, we conclude that, in a data-scarce scenario, or one in which there are very few adversarial examples of a certain type to learn from,

our approach would provide a strong initial defence against these examples until enough had been gathered to conduct effective traditional adversarial training with a large dataset. Another advantage of our approach is that a siamese verification network can be added to any existing CNN to verify any classifications, akin to two-factor authentication, so even when a model has been traditionally adversarially trained on a large amount of adversarial data, a siamese verification network would still be useful to verify that the model's prediction was correct.

Also, we have shown that it is possible to generate adversarial examples to attack siamese neural networks and reduce them to worse than random guessing levels in their predictions, as the accuracy of the face classification model drops below 50% when under attack, and 50% would be equivalent to random guessing in a binary classification problem such as this. Therefore, we conclude that our attack variants have the potential to be effective against siamese neural network models.

4.1.3 Limitations

The main limitation of the study is that, whilst we present strong empirical evidence to support our approach, it is difficult to provide an absolute proof that, in the context of one-shot learning, siamese verification networks are more effective than traditionally adversarially trained neural networks at providing adversarial robustness for every possible network. Indeed, we saw that the speech commands siamese verification network was slightly less effective than the other siamese networks, showing that effectiveness can vary, and our sample of three neural network models for comparison between approaches could be expanded on in the future. More variety in the neural network library could also be provided, such as adding a natural language model. Also, due to time constraints, we did not generate adaptive attacks to attack the defended neural networks, one of Carlini et al's recommendations [40], although this did not affect the aim of comparing the two approaches to initially defending a neural network with one-shot learning.

Finally, although we observed that our approach was significantly more robust than traditional adversarial training in one-shot learning, the accuracy values of ~70% on adversarial data suggests that these networks can be improved to achieve higher levels of robustness. Whilst we committed to building small neural networks in this study, a larger neural network architecture for the siamese verification networks may lead to greater adversarial robustness, although this increase in size would make formal verification of our architecture more challenging.

This project was also limited by a lack of computational resources, so it was not possible to generate larger models or datasets than what is present in the final code.

4.1.4 Recommendations

In terms of future research, there are several possible directions that may branch from this study. Firstly, it would be useful to test our approach against $L2$-norm adversarial attacks, expanding the threat model proposed in this study and making our defence more versatile. Moreover, the test performed in this study could be repeated on different neural network models, perhaps larger in size, which would demonstrate whether this approach can be applied to more complex problems. Additionally, the study could be expanded by developing adpative attacks that attack the defended neural networks, and then one-shot training models, both traditional and siamese, on these new examples to see if the difference in performance still remains.

Other future research could also adapt the architecture we provide for the siamese verification networks and increase the complexity to try and increase the robustness to greater levels than in our results. Alternatively, researchers may attempt to formally verify this approach, thus applying mathematical proof to reinforce our empirical evidence. We also hope that our 'LittleAdversary' library of pre-trained models can be used by researchers to develop new attacks and defences without having to spend time developing neural networks to attack or defend. More pretrained models could also be developed in the future, enabling researchers to develop their adversarial attacks and defences using realistic, security-critical adversarial examples.

4.2 Conclusion

Overall, we have presented a novel approach to attain a level of one-shot adversarial robustness for classification CNNs that is far greater than the current most popular method, adversarial training. We compared our siamese verification network method to traditional adversarial training in a one-shot setting and found that, not only was our approach significantly more robust to adversarial examples, it also had good levels of transferability to defend against attacks that the networks had not been trained for. There are, to the best of our knowledge, no other adversarial defences that apply one-shot learning, or that can be effective in a one-shot setting, so we hope that future research can expand upon this approach to produce fully robust siamese verification networks that attain similar robustness to an adversarially trained model trained on a large dataset consisting of thousands of adversarial examples.

Moreover, we have successfully adapted adversarial attacks from the CleverHans library to attack siamese neural networks, allowing for a more in-depth exploration into the potential vulnerabilities of siamese neural networks in future research.

Finally, we built the 'LittleAdversary' library of pre-trained neural networks that emulate security-critical tasks that can be used as a benchmark for researchers to test new attacks and defences against, as we demonstrated in this study. The library also includes our attack variants, our study, a pipeline to create siamese verification networks to defend the pretrained models, and a framework to evaluate defences. All of this functionality is written in Jupyter notebooks that give detailed instructions on how to use the library in order to to aid in future adversarial machine learning research.

References

[1] M. I. Jordan and T. M. Mitchell. "Machine learning: Trends, Perspectives, and prospects". In: *Science* 349.6245 (July 2015), pp. 255–260. DOI: 10.1126/science.aaa8415.

[2] Tom Michael Mitchell. *Machine learning*. McGraw-Hill, 1997.

[3] Jiuxiang Gu et al. "Recent advances in convolutional neural networks". In: *Pattern Recognition* 77 (2018), pp. 354–377. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2017.10.013. URL: https://www.sciencedirect.com/science/article/pii/S0031320317304120.

[4] S. Kevin Zhou et al. "Deep reinforcement learning in medical imaging: A literature review". In: *Medical Image Analysis* 73 (2021), p. 102193. ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2021.102193. URL: https://www.sciencedirect.com/science/article/pii/S1361841521002395.

[5] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2014. DOI: 10.48550/ARXIV.1412.6572. URL: https://arxiv.org/abs/1412.6572.

[6] Kevin Eykholt et al. *Robust Physical-World Attacks on Deep Learning Models*. 2017. DOI: 10.48550/ARXIV.1707.08945. URL: https://arxiv.org/abs/1707.08945.

[7] Battista Biggio and Fabio Roli. "Wild patterns: Ten years after the rise of adversarial machine learning". In: *Pattern Recognition* 84 (2018), pp. 317–331. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2018.07.023. URL: https://www.sciencedirect.com/science/article/pii/S0031320318302565.

[8] Anirban Chakraborty et al. "Adversarial attacks and defences: A survey". In: *arXiv preprint arXiv:1810.00069* (2018).

[9] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. "Siamese neural networks for one-shot image recognition". In: *ICML deep learning workshop*. Vol. 2. 1. Lille. 2015.

[10] Nicolas Papernot et al. "Technical report on the cleverhans v2. 1.0 adversarial examples library". In: *arXiv preprint arXiv:1610.00768* (2016).

[11] Anders Krogh. "What are artificial neural networks?" In: *Nature Biotechnology* 26.2 (2008), pp. 195–197. DOI: 10.1038/nbt1386.

[12] Yifang Ma et al. "Artificial intelligence applications in the development of autonomous vehicles: A survey". In: *IEEE/CAA Journal of Automatica Sinica* 7.2 (2020), pp. 315–329.

[13] Samuel G. Finlayson et al. "Adversarial attacks on medical machine learning". In: *Science* 363.6433 (2019), pp. 1287–1289. DOI: 10.1126/science.aaw4399. eprint: https://www.science.org/doi/pdf/10.1126/science.aaw4399. URL: https://www.science.org/doi/abs/10.1126/science.aaw4399.

[14] Christian Szegedy et al. *Intriguing properties of neural networks*. 2013. DOI: 10.48550/ARXIV.1312.6199. URL: https://arxiv.org/abs/1312.6199.

[15] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. "Adversarial examples in the physical world". In: *Artificial intelligence safety and security*. Chapman and Hall/CRC, 2018, pp. 99–112.

[16] Nir Morgulis et al. *Fooling a Real Car with Adversarial Traffic Signs*. 2019. DOI: 10.48550/ARXIV.1907.00374. URL: https://arxiv.org/abs/1907.00374.

[17] 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*. 2016, pp. 1528–1540.

[18] Anirban Chakraborty et al. *Adversarial Attacks and Defences: A Survey*. 2018. DOI: 10.48550/ARXIV.1810.00069. URL: https://arxiv.org/abs/1810.00069.

[19] Nathan Inkawich. *Adversarial example generation*. 2022. URL: https://pytorch.org/tutorials/beginner/fgsm_tutorial.html#:~:text=One%20of%20the%20first%20and%20most%20popular%20adversarial%20attacks%20to,is%20described%20by%20Goodfellow%20et..

[20] Nicolas Papernot et al. *Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks*. 2015. DOI: 10.48550/ARXIV.1511.04508. URL: https://arxiv.org/abs/1511.04508.

[21] Weilin Xu, David Evans, and Yanjun Qi. "Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks". In: *Proceedings 2018 Network and Distributed System Security Symposium*. Internet Society, 2018. DOI: 10.14722/ndss.2018.23198. URL: https://doi.org/10.14722%2Fndss.2018.23198.

[22] Nicholas Carlini and David Wagner. "Defensive Distillation is Not Robust to Adversarial Examples". In: (July 2016).

[23] Nina Narodytska and Shiva Kasiviswanathan. "Simple Black-Box Adversarial Attacks on Deep Neural Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2017, pp. 1310–1318. DOI: 10.1109/CVPRW.2017.172.

[24] Florian Tramèr et al. *Ensemble Adversarial Training: Attacks and Defenses*. 2017. DOI: 10.48550/ARXIV.1705.07204. URL: https://arxiv.org/abs/1705.07204.

[25] Davide Corsi, Enrico Marchesini, and Alessandro Farinelli. "Formal verification of neural networks for safety-critical tasks in deep reinforcement learning". In: *Uncertainty in Artificial Intelligence*. PMLR. 2021, pp. 333–343.

[26] Nicolas Papernot et al. "Technical Report on the CleverHans v2.1.0 Adversarial Examples Library". In: *arXiv preprint arXiv:1610.00768* (2018).

[27] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[28] Daniel Smilkov Smilkov and Shan Carter. *Tensorflow - Neural Network Playground*. 2016. URL: https://playground.tensorflow.org/.

[29] Jane Bromley et al. *Signature verification using a "Siamese" time delay neural network. Series in Machine Perception and Artificial Intelligence. 1994; 25–44*. 1994.

[30] Davide Chicco. "Siamese neural networks: An overview". In: *Artificial neural networks* (2021), pp. 73–94.

[31] Mohammad Shorfuzzaman and M Shamim Hossain. "MetaCOVID: A Siamese neural network framework with contrastive loss for n-shot diagnosis of COVID-19 patients". In: *Pattern recognition* 113 (2021), p. 107700.

[32] James Loy. *Neural network projects with python*. URL: https://www.oreilly.com/library/view/neural-network-projects/9781789138900/7cf02853-460a-4fcd-9cc6-cd698bc5978e.xhtml.

[33] Cyrus R Mehta and Nitin R Patel. "IBM SPSS exact tests". In: *Armonk, NY: IBM Corporation* (2011), pp. 23–24.

[34] Thomas Kluyver et al. *Jupyter Notebooks-a publishing format for reproducible computational workflows.* Vol. 2016. 2016.

[35] Yann LeCun and Corinna Cortes. "MNIST handwritten digit database". In: (2010). URL: http://yann.lecun.com/exdb/mnist/.

[36] Johannes Stallkamp et al. "The German Traffic Sign Recognition Benchmark: A multi-class classification competition". In: *The 2011 International Joint Conference on Neural Networks*. 2011, pp. 1453–1460. DOI: 10.1109/IJCNN.2011.6033395.

[37] Pete Warden. *Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition*. 2018. arXiv: 1804.03209 [cs.CL].

[38] F.S. Samaria and A.C. Harter. "Parameterisation of a stochastic model for human face identification". In: *Proceedings of 1994 IEEE Workshop on Applications of Computer Vision*. 1994, pp. 138–142. DOI: 10.1109/ACV.1994.341300.

[39] Nicholas Carlini and David Wagner. *Towards Evaluating the Robustness of Neural Networks*. 2017. arXiv: 1608.04644 [cs.CR].

[40] Nicholas Carlini et al. *On Evaluating Adversarial Robustness*. 2019. arXiv: 1902.06705 [cs.LG].

[41] Henry B Mann and Donald R Whitney. "On a test of whether one of two random variables is stochastically larger than the other". In: *The annals of mathematical statistics* (1947), pp. 50–60.

[42] Giovanni Di Leo and Francesco Sardanelli. "Statistical significance: p value, 0.05 threshold, and applications to radiomics—reasons for a conservative approach". In: *European radiology experimental* 4.1 (2020), pp. 1–8.

[43] Guillermo Macbeth, Eugenia Razumiejczyk, and Rubén Daniel Ledesma. "Cliff's Delta Calculator: A non-parametric effect size program for two groups of observations". In: *Universitas Psychologica* 10.2 (2011), pp. 545–555.

[44] Aleksander Madry et al. *Towards Deep Learning Models Resistant to Adversarial Attacks*. 2019. arXiv: 1706.06083 [stat.ML].

[45] Yinpeng Dong et al. *Boosting Adversarial Attacks with Momentum*. 2018. arXiv: 1710.06081 [cs.LG].

[46] Samuel Sanford Shapiro and Martin B Wilk. "An analysis of variance test for normality (complete samples)". In: *Biometrika* 52.3/4 (1965), pp. 591–611.

[47] Brian B Schultz. "Levene's test for relative variation". In: *Systematic Zoology* 34.4 (1985), pp. 449–456.

[48] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[49] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[50] Oct. 2022. URL: https://www.upgrad.com/blog/basic-cnn-architecture/.

[51] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[52] Cassiebreviu. *Introduction to audio classification with tensorflow - training*. URL: https://learn.microsoft.com/en-us/training/modules/intro-audio-classification-tensorflow/?WT.mc_id=api_CatalogApi.

[53] Sudharsan Ravichandiran. *Hands-On Meta Learning with Python: Meta Learning Using One-Shot Learning, MAML, Reptile, and Meta-SGD with TensorFlow*. Packt Publishing Ltd, 2018.

Acknowledgments

I would like to acknowledge my supervisor, Dr. Achim Brucker, for all the support that he has provided me throughout the process of producing this thesis.