WSD030 Advanced Project; Research Report

Name:	Janith Thanthulage	ID Number: B523685
Degree Programme:	Electronic and Electrical Engineering (WSUM10)	
Project Title:	Robust Image Classification using Deep Learning	

How to complete the report

(a) Just add your text in the places indicated with "*your text here*"; leave the headings and descriptions in place to remind your assessors what you have been told to do.

(b) There is no page limit but common sense is expected.

(c) The report should be submitted electronically in a single file in Portable Document Format (.pdf) form via the WS25META pages on Learn. The filename should be your ID number and *research* (hyphenated) with the *.pdf* extension, for example A123456-research.pdf. Your name will be associated with the file by Learn.

Project Description: *A definitive description of your project. This can be an expansion of the original description or entirely new but it should describe the project that you are actually doing.*

The rise of machine learning and data-driven artificial intelligence in the area of pattern/object recognition has found incredibly accurate results. Hence, it has found itself being used in many safety and security critical tasks such as in self-driving cars and malware detection software. Therefore, it is pertinent that these systems be robust and trained to withstand malicious attempts at disruption from adversarial examples. This project will focus on the adversarial machine learning methods that exist to train a robust model.

Project Aim: *A single definitive project aim; it should state in one sentence what are you trying to do.*

To develop and analyse deep learning techniques for image classification in the presence of adversarial attacks.

Project Objectives: *What do you need to do to satisfy the aim? Objectives should be simple and measureable; ask yourself the question can I decide if I have achieved this or not? Note that personal objectives are not relevant; "learning to use a soldering iron" is not a project objective.*

To understand and develop deep learning based image classification using an appropriate scientific software.

To characterise various forms of attacks on image classification algorithms and develop a robust classification through an adversarial learning framework.

To device appropriate simulation setup to evaluate performance of classification algorithms

To produce a technical report on the impact of adversarial attacks on deep learning classification systems and the defences used against them.

Project Deliverables: *What do you need to produce to meet the objectives; this can be software, hardware or even the exposition of a theory. This should not include the written deliverables for the project, those are understood.*

In order to meet the above listed objectives, a convolution neural network, capable of performing classification on a well-known dataset, will need to be developed to a good accuracy.

Adversarial examples capable of causing misclassifications on the said classifier will then be developed: this misclassification will need to be demonstratable.

Lastly, a separate classifier capable of withstanding these adversarial attacks will need to be developed. This classifier will require a demonstratable improvement in classification.

Research Sources: *List (using the format specified for the final paper) all of the sources of information that you have used in your research.*

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–11, 2015.

[2] A. Mikołajczyk and M. Grochowski, “Data augmentation for improving deep learning in image classification problem,” *2018 Int. Interdiscip. PhD Work. IIPhDW 2018*, pp. 117–122, 2018.

[3] A. Sotgiu *et al.*, “Deep Neural Rejection against Adversarial Examples,” 2019.

[4] B. Biggio and F. Roli, “Wild patterns: Ten years after the rise of adversarial machine learning,” *Pattern Recognit.*, vol. 84, pp. 317–331, 2018.

[5] Y. Geifman and R. El-Yaniv, “SelectiveNet: A Deep Neural Network with an Integrated Reject Option,” 2019.

[6] A. B. C. Lassifiers and A. Authors, “Knows when it doesn’t know: Deep Abstaining Classifiers,” vol. 33, no. 4, pp. 899–943, 1968.

[7] M. Melis, A. Demontis, B. Biggio, G. Brown, G. Fumera, and F. Roli, “Is Deep Learning Safe for Robot Vision? Adversarial Examples against the iCub Humanoid,” 2017.

[8] O. Poursaeed, I. Katsman, B. Gao, and S. Belongie, “Generative Adversarial Perturbations,” 2018.

[9] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, “Generative Adversarial Networks: An Overview,” *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 53–65, 2018.

[10] K. Davaslioglu and Y. E. Sagduyu, “Generative adversarial learning for spectrum sensing,” *IEEE Int. Conf. Commun.*, vol. 2018-May, pp. 1–6, 2018.

[11] A. Bendale and T. E. Boult, “Towards open set deep networks,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 1563–1572, 2016.

[12] H. Kwon, Y. Kim, H. Yoon, and D. Choi, “Fooling a Neural Network in Military Environments: Random Untargeted Adversarial Example,” *Proc. - IEEE Mil. Commun. Conf. MILCOM*, vol. 2019-Octob, pp. 456–461, 2019.

[13] F. Ranzato and M. Zanella, “Robustness Verification of Support Vector Machines,” pp. 271–295, 2019.

[14] A. Demontis *et al.*, “Why Do Adversarial Attacks Transfer? Explaining Transferability of Evasion and Poisoning Attacks,” 2018.

[15] B. Biggio, B. Nelson, and P. Laskov, “Poisoning attacks against support vector machines,” *Proc. 29th Int. Conf. Mach. Learn. ICML 2012*, vol. 2, pp. 1807–1814, 2012.

[16] N. Carlini and D. Wagner, “Towards Evaluating the Robustness of Neural Networks,” *Proc. - IEEE Symp. Secur. Priv.*, pp. 39–57, 2017.

[17] N. Papernot, P. Mcdaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, “The limitations of deep learning in adversarial settings,” *Proc. - 2016 IEEE Eur. Symp. Secur. Privacy, EURO S P 2016*, pp. 372–387, 2016.

[18] B. Biggio *et al.*, “Evasion attacks against machine learning at test time,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8190 LNAI, no. PART 3, pp. 387–402, 2013.

[19] C. Xiao, B. Li, J. Y. Zhu, W. He, M. Liu, and D. Song, “Generating adversarial examples with adversarial networks,” *IJCAI Int. Jt. Conf. Artif. Intell.*, vol. 2018-July, pp. 3905–3911, 2018.

[20] J. Su, D. V. Vargas, and K. Sakurai, “One Pixel Attack for Fooling Deep Neural Networks,” *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 828–841, 2019.

[21] N. Papernot, I. Goodfellow, M. Abadi, K. Talwar, and Ú. Erlingsson, “Semi-supervised knowledge transfer for deep learning from private training data,” *5th Int. Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc.*, no. 2015, pp. 1–16, 2019.

[22] A. Liu *et al.*, “Perceptual-Sensitive GAN for Generating Adversarial Patches,” *Proc. AAAI Conf. Artif. Intell.*, vol. 33, pp. 1028–1035, 2019.

[23] K. Eykholt *et al.*, “Robust Physical-World Attacks on Deep Learning Models,” 2017.

Influential Sources: *Pick from your research sources those that have had most influence on your thinking and/or the direction of your project and explain why. Make sure that you cite and/or quote your source material correctly.*

There are a select number of resources from the above mentioned that have held more influence on the direction of the project.

[3] A. Sotgiu *et al.*, “Deep Neural Rejection against Adversarial Examples,” 2019.

It was intended for this project to build upon the previously held experience and knowledge of developing CNNs (convolution neural networks) for visual recognition tasks. Given this previous knowledge, the weaknesses of neural networks were also known to a degree. It was known how regardless of how good the training of the deep neural network is, the ability to mislead them has always been present. Based on this

knowledge, the project was directed towards conducting research on techniques used to best defend against such attempts.

[3] was the first conducted research into defences against adversarial examples (an instance with a feature perturbation to intentionally cause false predictions). It describes a method of deep neural rejection as a potential answer to the threats to security and general classification posed by adversarial examples. It effectively poses a method to reject adversarial items prior to classification hence reducing the number of misclassifications and improving the robustness of the model to such threats.

Only evasion-type attacks were used to test the model in this research source. Due to its promise, it is intended that this proposed architecture be tested against other adversarial examples such as the training-time poisoning attacks it mentions for possible future work. From this, dependant on the results of various testing, it will be possible to solidify the method as an effective defence, or alternatively offer an improvement upon it.

[20] J. Su, D. V. Vargas, and K. Sakurai, "One Pixel Attack for Fooling Deep Neural Networks," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 828–841, 2019.

&

[22] A. Liu *et al.*, "Perceptual-Sensitive GAN for Generating Adversarial Patches," *Proc. AAAI Conf. Artif. Intell.*, vol. 33, pp. 1028–1035, 2019.

[20] also illustrated some interesting research into causing false classifications by only altering a single pixel on an image. It demonstrates the fundamental inability in current DNNs (Deep Neural Networks) to effectively ignore small adversarial perturbations. It states that noise reduction pre-processing techniques have been utilized to defend against such attacks. However, by adding this extra processing layer, the response rate of real-time detection systems will be inefficient making the method of defence less than ideal. The method proposed in [3] offers a far faster response however, remains untested against such perturbation based attacks. It is of particular interest to perform this test at a later point in the project.

Taken further, [22] explores a similar topic of interest, where instead of altering a single pixel, a 'patch' of pixels is artificially generated using a GAN and consequently inserted onto an image to simulate the effect of 'real world noise' on an object being classified such as dirt and stickers. The testing of the developed defence against such patches would be a desirable added comparison to make if time will allow for an interesting analysis of performance under real-world conditions.

Essentially both these adversarial examples expose the problem of current neural networks being unable to detect and ignore adversarial perturbation without severe additional pre-processing. It is a very interesting problem to further pursue and compare against the model proposed in [3].

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–11, 2015.

&

[23] K. Eykholt *et al.*, "Robust Physical-World Attacks on Deep Learning Models," 2017.

[1] is a key source of reference for understanding an adversarial example at its core. It details a very effective method of creating adversarial examples (referred to in the paper as the 'fast gradient sign method'). In addition to describing how to construct adversarial examples from given trained data, [1] also clearly exposes the vulnerabilities of deep networks which adds support to the importance of the direction of the project.

This threat is further supported by [23] which further reveals the threat posed by such adversarial examples by demonstrating recognition systems such as those used in self-driving cars misclassifying road signs. The experiments discussed were conducted in both static and dynamic environments. Objects were used to camouflage street signs similar to the generated adversaries in [22] and clearly demonstrated the reduction in classification accuracy to the point of misclassification in certain instances.

Nevertheless, the 'family of fast methods for generating adversarial examples' (discussed in [1]) is what is of most use. It also demonstrates a type of 'white-box' adversarial attack where a layer of noise can be added to an image causing widely used and known classifiers to drastically alter their classification of the image. It

would be desirable to test the deep neural rejection model proposed in [3] against this simple but effective method.

Current Project State: *Explain clearly where you are in your project and what you have done to date.*

The Keras library was used as it allows for easy addition of layers and is again very widely used for its fast implementation. In this early stage of the project, the ability to quickly test different architectures was thought to be important. If possible, it would be desirable to develop a network without the use of this library. One advantage of such an implementation (in addition to gaining a greater understanding) would be the ability to easily run a model on any PC with python and the basic NumPy library installed.

Using Keras, a convolutional neural network (CNN) capable of performing classification (to a good accuracy) on both the MNIST and Cifar10 datasets has been built and tested. These datasets were chosen due to both being well known and widely used in the community. This is further supported by their use in many of the references listed [1, 3, 4, 6]. In addition to being convenient to obtain, it also allows for easy comparison between known models. The model used for classification of the MNIST dataset is illustrated in Figure 1. This architecture used is the same as the MNIST model used in [3] (used for easy comparison of results between implementations).

Figure 2 illustrates the accuracy and loss graphs for the MNIST model in relation to the iterations. The cross entropy loss graph in particular is very helpful in identifying a well fitted model to the data. Both the training loss and testing (validation) loss can be seen to decrease to a point of stability with a very small disparity between the two by the last epoch. The same hyperparameters mentioned in [3] were used for the training of the model:

- Learning rate = 0.1
- Momentum = 0.9
- Dropout = 0.5
- Batch size = 128
- Epochs = 50

This was again used for comparison purposes.

```
In [9]: cnn_model.summary()
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 28, 28, 32)        320
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 28, 28, 32)        9248
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
dropout_4 (Dropout)          (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 14, 14, 64)        18496
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 14, 14, 64)        36928
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
dropout_5 (Dropout)          (None, 7, 7, 64)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 3136)              0
_________________________________________________________________
dense_6 (Dense)              (None, 200)               627400
_________________________________________________________________
dense_7 (Dense)              (None, 200)               40200
_________________________________________________________________
dense_8 (Dense)              (None, 10)                2010
=================================================================
```

Figure 1 – CNN Architecture for classification of MNIST dataset.

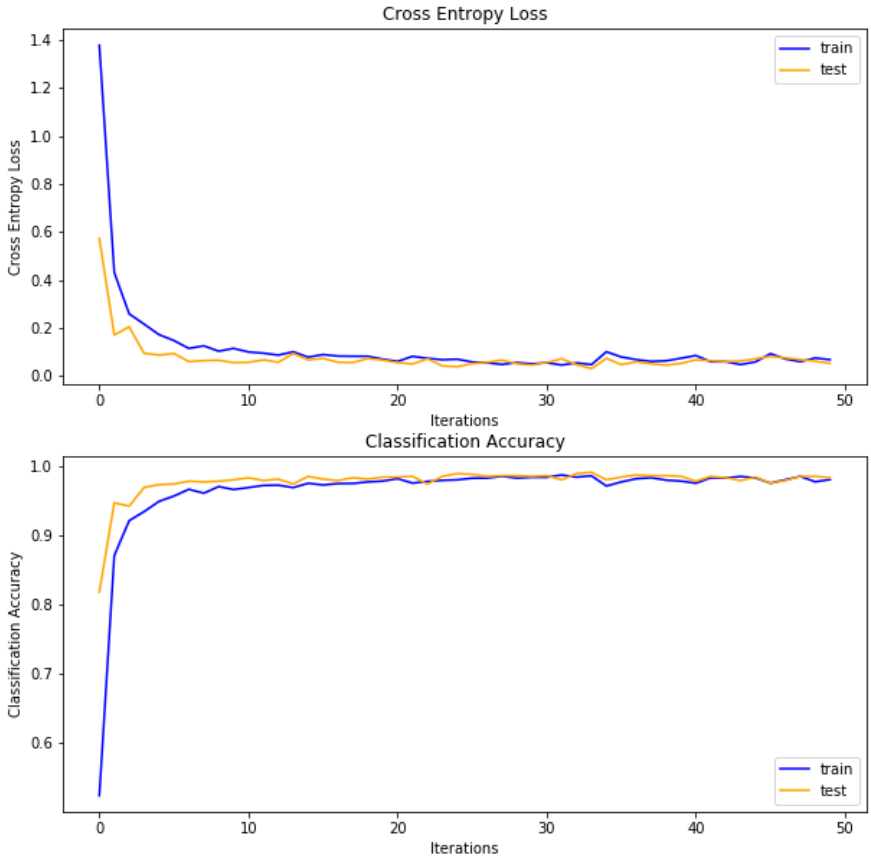

Figure 2 – Loss and Accuracy Graphs for CNN used to classify MNIST

Due to the work illustrated in [22], there was some interest in the functionality of GANs (Generative Adversarial Networks). Its ability to generate images in the hope of being able to generate adversarial perturbed examples as described in [22] was consequently tested. To better understand its functionality and gauge its further use, a GAN to generate 'picture-like' images based off the cifar10 dataset was developed.

This again made use of the Keras library for the quick deployment of models that it allowed. To train a realistic generator to generate artificial images, the network consists of two models: a discriminator (discriminating between real and generated images) and a generator (generating artificial images). It consisted of the structure illustrated in Figure 3.

The discriminator model follows a similar structure to a conventional CNN however has 1 output classification: Real (1) or Fake/Generated (0). The discriminator model (see Figure 4) was first trained for a batch of real images taken from the cifar10 dataset for a given epoch. It was then consequently trained for an equal sized batch of generated images (labelled Faked/0). In response to the loss score of the discriminator, the generator (see Figure 5) weights would then be appropriately updated.

The model was trained for a total of 200 epochs. The improvement in realistically generated images can be clearly seen when comparing between the 10th (Figure 6) and 200th (Figure 7) epochs. The knowledge gained from this exercise was very valuable towards better understanding the techniques suggested in [22]. It would be desirable to generate similar adversarial examples mentioned in addition to others such as poisoning, evasion and 'fast adversarial example generation' described in [4], [3] and [1] relatively.

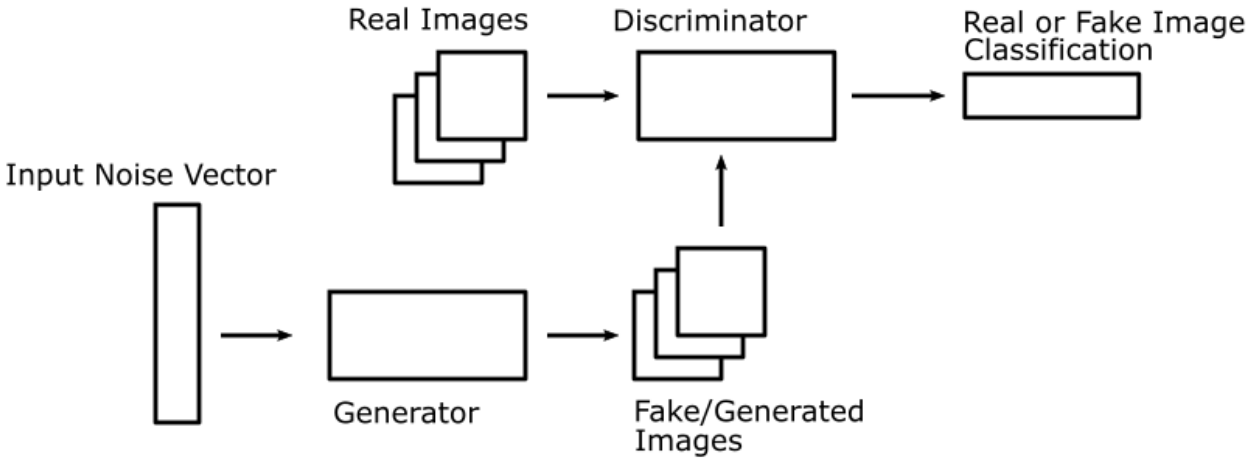

Figure 3 – Training Structure of a GAN

```
In [7]: dis_model.summary()
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 32, 32, 64)        1792
_________________________________________________________________
leaky_re_lu_8 (LeakyReLU)    (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 16, 16, 128)       73856
_________________________________________________________________
leaky_re_lu_9 (LeakyReLU)    (None, 16, 16, 128)       0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 8, 8, 128)         147584
_________________________________________________________________
leaky_re_lu_10 (LeakyReLU)   (None, 8, 8, 128)         0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 4, 4, 256)         295168
_________________________________________________________________
leaky_re_lu_11 (LeakyReLU)   (None, 4, 4, 256)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 4096)              0
_________________________________________________________________
dropout_3 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 4097
=================================================================
```

Figure 4 – Discriminator Model Architecture

```
In [8]: gen_model.summary()
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_6 (Dense)              (None, 4096)              413696
_________________________________________________________________
leaky_re_lu_12 (LeakyReLU)   (None, 4096)              0
_________________________________________________________________
reshape_1 (Reshape)          (None, 4, 4, 256)         0
_________________________________________________________________
conv2d_transpose_3 (Conv2DTr (None, 8, 8, 238)         975086
_________________________________________________________________
leaky_re_lu_13 (LeakyReLU)   (None, 8, 8, 238)         0
_________________________________________________________________
conv2d_transpose_4 (Conv2DTr (None, 16, 16, 238)       906542
_________________________________________________________________
leaky_re_lu_14 (LeakyReLU)   (None, 16, 16, 238)       0
_________________________________________________________________
conv2d_transpose_5 (Conv2DTr (None, 32, 32, 238)       906542
_________________________________________________________________
leaky_re_lu_15 (LeakyReLU)   (None, 32, 32, 238)       0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 32, 32, 3)         6429
=================================================================
```

Figure 5 – Generator Model Architecture

Figure 6 – Generated Images at 10 epochs

Figure 7 – Generated Images at 200 epochs

Currently the state of the project, as illustrated by the Gantt chart in FIGURE, is directed towards using the modelled MNIST classification CNN to include the aspects of deep neural rejection mentioned in [3]. As part of this process, time has been invested into researching the development and use of Support Vector Machines (SVMs) so that the deep neural rejection architecture can be emulated in testing. Due to individual layers of the MNIST CNN having to be used to emulate this architecture, transfer learning was also researched as part of this process. Once this architecture is complete, adversarial examples can then be developed (similar to those previously mentioned) to then test the defence model effectiveness against such attacks.

Activity Breakdown: *List the major activities that you have identified as necessary for the successful completion of your project. In each case specify the status as complete, current, pending or desirable, and the date that they were or are expected to be completed.*

A basic breakdown of activities completed, pending and desirable are listed below. A full breakdown of tasks can be found in Figure 8.

- Research basic implementations of CNNs – COMPLETED 06/10/19
- Implement CNN on Cifar10 dataset – COMPLETED 13/10/19
- Implement CNN from scratch on a dataset – DESIRABLE
- Implement CNN on MNIST dataset – COMPLETED 17/11/19
- Research adversarial learning techniques – COMPLETED 13/10/19
- Research Generative Adversarial Network (GAN) implementations – COMPLETED 17/11/19
- Implement CNN architecture mentioned in [3] to a 90+% accuracy – COMPLETED 24/11/19
- Given the CNN architecture, train an SVM for the individual layers of the created CNN specified in [3] – CURRENT 01/01/20
- Combine the 3 separate SVMs into a combined RBF SVM – PENDING 05/01/20
- Use the above architecture to develop the Deep Neural Rejection architecture in [3] - PENDING 02/02/20
- Research further into Adversarial Attack methods such as poisoning and evasion – PENDING 23/02/20
- Experiment the effect of Deep Neural Rejection at defending against adversarial attacks mentioned in [3] – PENDING 08/03/20
- Compare Deep Neural Rejection to other forms of rejection networks – DESIRABLE
- Compare the effectiveness of Deep Neural Rejection at defending against perceptual patches - DESIRABLE
- Compare the effectiveness of Deep Neural Rejection to the ‘fast gradient sign’ adversaries – DESIRABLE

Risk Assessment: *List the risks associated with the successful completion of your project, your assessment of their severity and likely effects and your suggestions for their mitigation.*

For a list of identified risks that would impact the completion of this project, see Table 1. See Table 2 for the associated risk matrix.

Appendix A – Gantt Chart of project plan

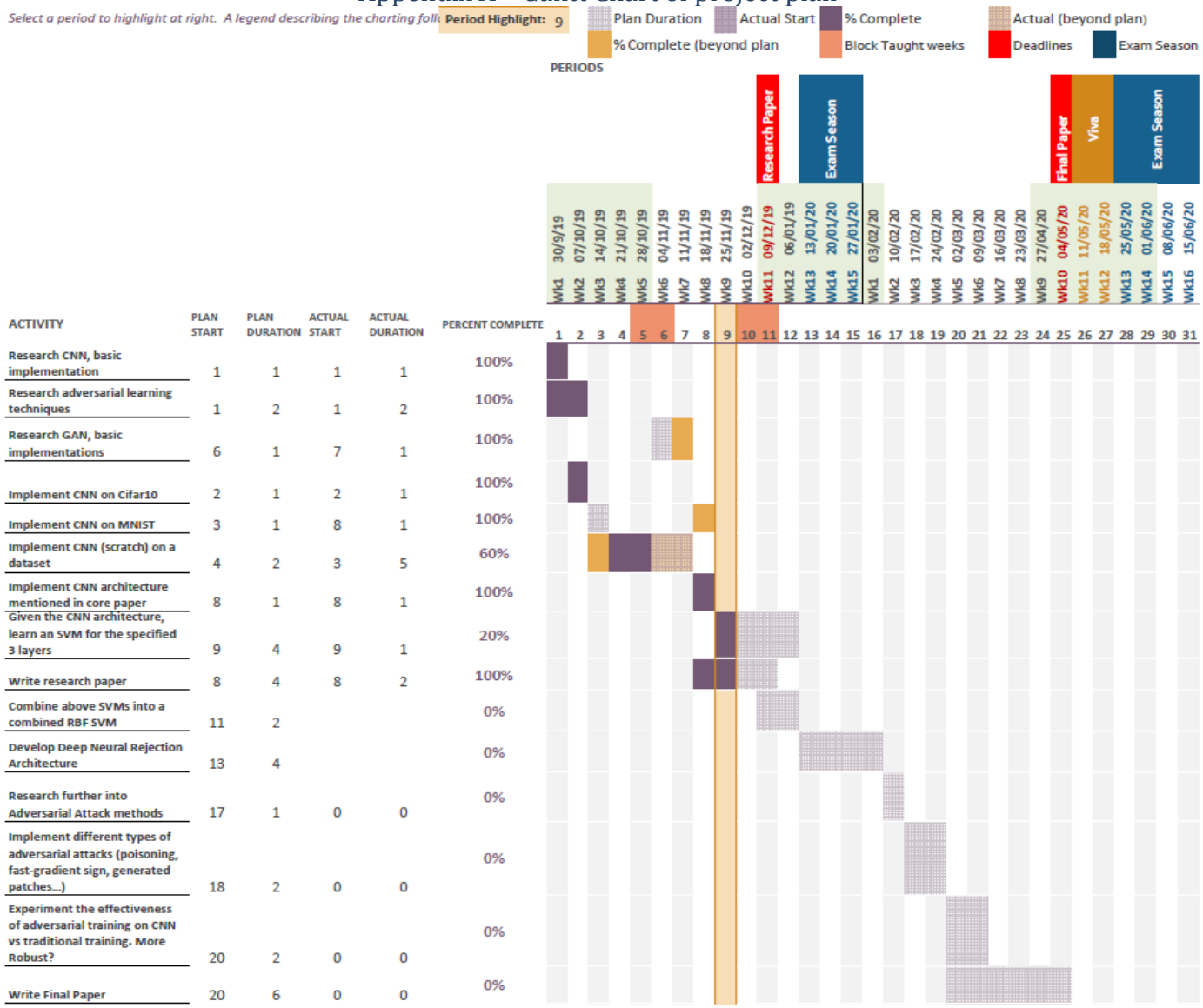

Figure 8 – Gantt Chart

Note: this does not take into account the weeks available during holidays.

Appendix B – Risk Assessment

Table 1 – Assessment and Mitigation of Risks

	Pre Mitigation			Post Mitigation				
Risk	Likelihood (L)	Impact (I)	Severity Score (S)	Likelihood (L)	Impact (I)	Severity Score (S)	Likely effect of risk occurring	Method of mitigation
Repetitive Strain Injury	3	4	12	1	3	3	Will impact the efficiency at which work can be conducted.	Taking regular breaks to avoid likelihood and stretching hands to additional mitigate the impact if to occur
PC data loss	2	5	10	2	1	2	Loss of any unsaved work and a delay proportional to the amount of work lost.	Perform regular backups onto an external hard drive and a cloud storage system to avoid data loss
Demonstration is not possible on the day of the viva	3	5	10	1	5	5	Will result in being unable to present much of the findings of the work completed.	Save all working versions of models on multiple sources. Avoid working on anything to do with project a week prior to the viva. Set up demonstration the day prior and ensure that setup is working every 2 days on the lead up to the viva.
Not able to develop a CNN model capable of good identification accuracy	1	5	5	1	1	1	Will lead to not being able to generate a good comparison between existing methods of defence.	Due to prior knowledge, this is unlikely to be the case. However, to absolutely mitigate against this risk, research into existing architecture will be conducted to maximise the chance of being able to recreate a similar model.

Not able to develop a method of creating adversarial examples	3	5	15	1	5	5	Unable to characterise various forms of adversarial attacks. Will then also lead to not being able to fully develop a robust method of defence against such attacks.	Due to a lack of prior knowledge, sufficient research will need to be performed. Attempts at developing adversarial examples will also need begin at an early stage in the project. This is key as to allow for good time to complete this key milestone.
Unable to develop a good method of robust classification	5	5	25	2	3	6	Unable to complete the topic of this project.	This risk exists due to a lack of prior knowledge on the subject. By performing prior research, the understanding of developing methods of defence methods should increase. With this, the likelihood of this risk will be reduced. Pre-existing, well documented and recreatable methods of defence against adversarial examples will also be used in order to reduce the impact of this risk.
Update in libraries causes errors throughout program resulting in inability to run.	2	5	10	1	3	3	Unable to demonstrate functionality	This risk exists due to updates in software that render other library components

Appendix C – Risk Matrix and Key

Table 2 – Risk Matrix

		Impact				
		1 – cause a delay of 1 day	2 – cause a delay of 1-2 days	3 – cause a delay of <week	4 – cause a delay of 1 week to 1 month	5 – cause a delay of 1+ month
Likelihood	1 – very rare, has never before occurred	1	2	3	4	5
	2 – rare, very low chance to occur	2	4	6	8	10
	3 – possible, has occurred previously	3	6	9	12	15
	4 – probable, has occurred multiple times a year	4	8	12	16	20
	5 – likely, has occurred multiple times a month	5	10	15	20	25

Key	
	A risk that puts the project at risk of not being completed. 30-100% of project not completed
	A risk that impacts the quality of the project. Little if any impact to the completion of the project. 0-30% of project not completed.
	A risk that has slight, if any impact to the quality of the project. No impact to the completion of the project.