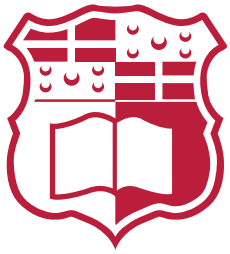

L-Università ta' Malta
Faculty of Information &
Communication Technology

# Advanced Computer Vision Individual Assignment

Matthias Bartolo

B.Sc. It (Hons) Artificial Intelligence (Third Year)

| | |
|---|---|
| Study Unit Code: | **ARI3129** |
| Study Unit: | **Advanced Computer Vision for Artificial Intelligence** |
| Lecturer: | **Dr Dylan Seychell** |
| Date: | **17 October 2023** |

# Contents

**Contents** i

**1 Neural Networks & TensorFlow Playground** 1
- 1.1 Task 1: Understanding Activation Functions . . . . . 1
- 1.2 Task 2: Exploring Learning Rates . . . . . 1
- 1.3 Task 3: Playing with Layers and Neurons . . . . . 2
- 1.4 Task 4: Optimisation Algorithms . . . . . 2
- 1.5 Conclusion . . . . . 3

**References** 4

**A Understanding Activation Functions** 5

**B Exploring Learning Rates** 8

**C Playing with Layers and Neurons** 9

**D Optimisation Algorithms** 10

**Plagiarism Form** 13

# 1 Neural Networks & TensorFlow Playground

Please note that the observations in this document, were inputted into ChatGPT to verify their accuracy. The prompts utilised, can be accessed through the following link: https://chat.openai.com/share/6ba74944-49dc-419c-a8c1-e623ad7cbfdd

## 1.1 Task 1: Understanding Activation Functions

The created neural network with one hidden layer utilised for this experiment [1], and which can be seen in Figure A.1 has the following properties:

- Hidden Layer consisting of four Neurons
- Learning Rate set to 0.03
- Problem Type set to Classification
- Regularisation set to None
- Dataset set to Circle

From the results in Tables: A.1, A.2, A.4, and A.3, it was noted that generally ReLU is the best performing function which enables the neural network to converge the fastest on the given datasets. Additionally, it was also noted that Tanh was the second best performing function, followed by Sigmoid, then Linear. It was generally noted, that the best three activation functions always converged, except for the instance of the Spiral dataset, where neither of the models could fit the dataset, due to the fact that only a single hidden layer is being used. Moreover, it was also noted that the Linear activation function did not converge for the Circle and Exclusive or datasets, as the model could not fit the data. However, this was not the case for the Gaussian dataset, as all of the activation functions managed to enable the created neural network to converge.

## 1.2 Task 2: Exploring Learning Rates

The pre-set neural network architecture was set to have the following properties:

- The default number of hidden layers was set to two
- Activation Function set to ReLU
- Problem Type set to Classification
- Regularisation set to None
- Dataset set to Circle

From the results obtained in Table B.1, it was noted that learning rates between 0.1 and 1 present the optimal learning rates given the specified scenario and Circle dataset. Additionally, it was also noted that high learning rates, which were larger than 1, tended to overshoot and display oscillations in the output graph. Moreover, it was

also noted that a small learning rate such as 0.0001 and high learning rates such as 10 did not enable the neural network to converge.

## 1.3 Task 3: Playing with Layers and Neurons

The pre-set neural network architecture was set to have the following properties:

- The learning rate was set to 0.01
- Activation Function set to ReLU
- Problem Type set to Classification
- Regularisation set to None
- Dataset set to Circle

From the results obtained in Table C.1, it was noted that with less hidden layers and neurons, the decision boundary is simple and may struggle to converge when trying to learn complex data patterns. On the other hand, a large amount of hidden layers and neurons in the neural network, would enable the model's capacity to learn complex data patterns, resulting in a decision boundary that is more complex and can better suit the training data. Additionally, it was also noted that with more layers and neurons, the number of epochs required to converge is less than when compared to the same neural network with less neurons and layers. Furthermore, when only one neuron was utilised, the decision boundary was only a straight line; therefore, more neurons and hidden layers are needed to make decision borders more complicated.

## 1.4 Task 4: Optimisation Algorithms

The neural network which was used for this experiment, consisted of two fully connected hidden layers, each having the ReLU activation functions, and the softmax activation function for output. Additionally, to facilitate comparisons in this section, the following settings were set to be constant during execution:

- Batch Size set to 8
- L2 Decay set to 0.001
- Learning Rate set to 0.01
- Activation Function set to ReLU

The following optimization methods were evaluated: SGD, SGD with Momentum (0.91), Adagrad, Adagrad with Momentum (0.91), Windowgrad, Windowgrad with Momentum (0.91), Adadelta, Adadelta with Momentum (0.91), Nesterov, and Nesterov with Momentum (0.91). The experiment was carried out on [2] and the input code which was used for such experiment is illustrated in code listing D.1.

From the output graphs displayed in Figures D.1, D.2 and D.3 it was noted that the best performing optimisation function was **Adadelta**. This was concluded as both the Adadelta and Adadelta with momentum functions had the best training accuracy, testing accuracy and least loss, when compared to the other functions. Additionally, it was also noted that given momentum, the optimisation functions converge much quicker than those without, when compared to their counterparts such as **SGD**.

## 1.5 Conclusion

In summary, this study has provided a comprehensive understanding of how a neural network could be fine-tuned. Modifying parameters such as learning rates and activation functions might significantly impact the model's performance, as achieving a balance between model complexity and efficiency is crucial. Selecting the right tool is important when it comes to factors such as activation functions or the optimisation functions.

# References

[1] Google Research. “Tensorflow playground.” (2016), [Online]. Available: https://playground.tensorflow.org/.

[2] A. Karpathy. “Convnetjs: Demonstrations.” (2022), [Online]. Available: https://cs.stanford.edu/people/karpathy/convnetjs/demo/trainers.html.

# Appendix A Understanding Activation Functions

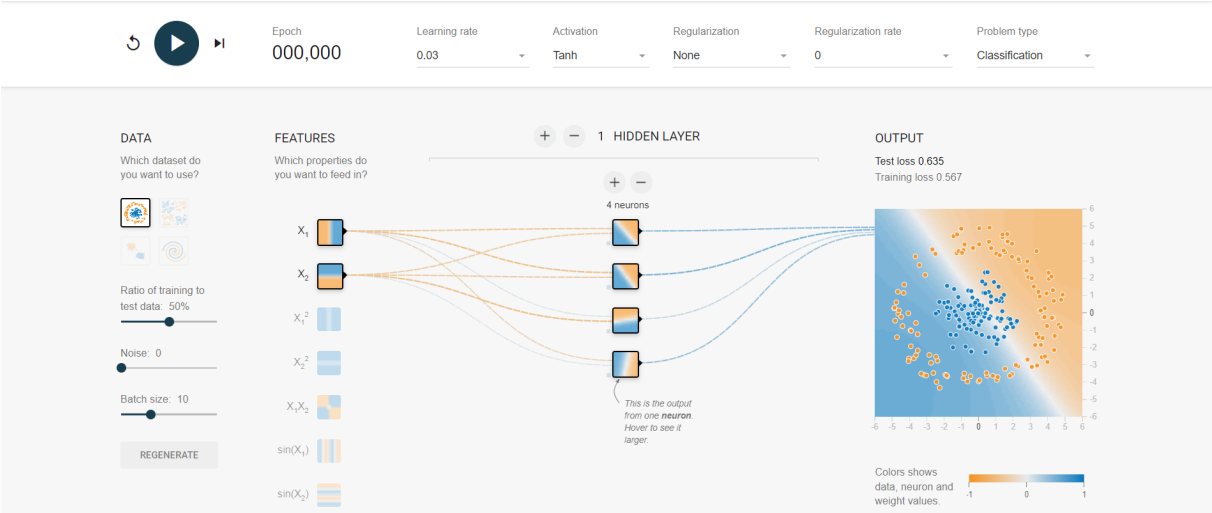


Figure A.1 Showing the created neural network with one hidden layer.

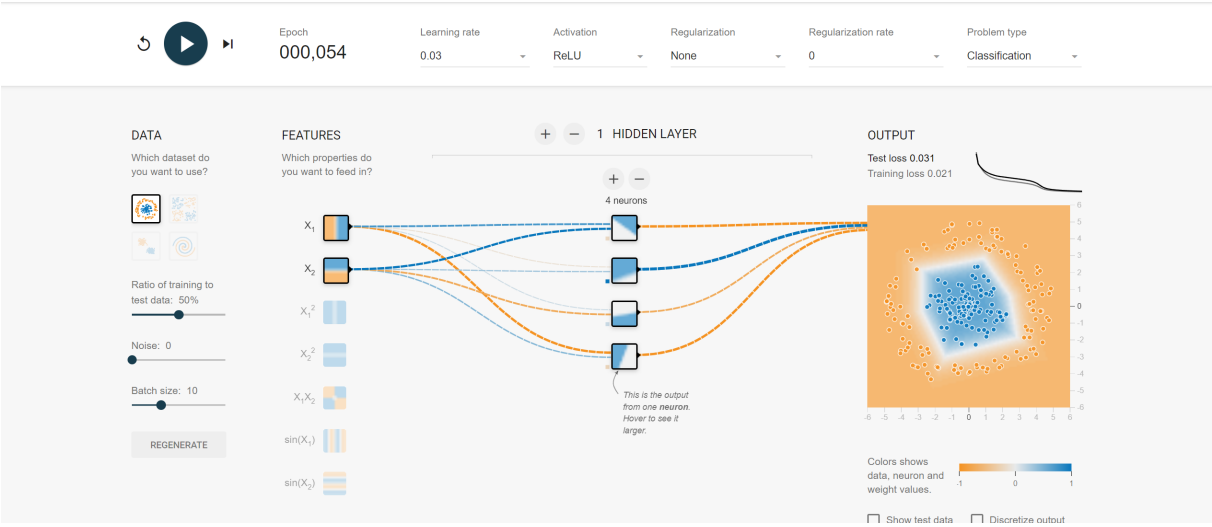


Figure A.2 Showing the created neural network which learned to fit the Circle dataset.

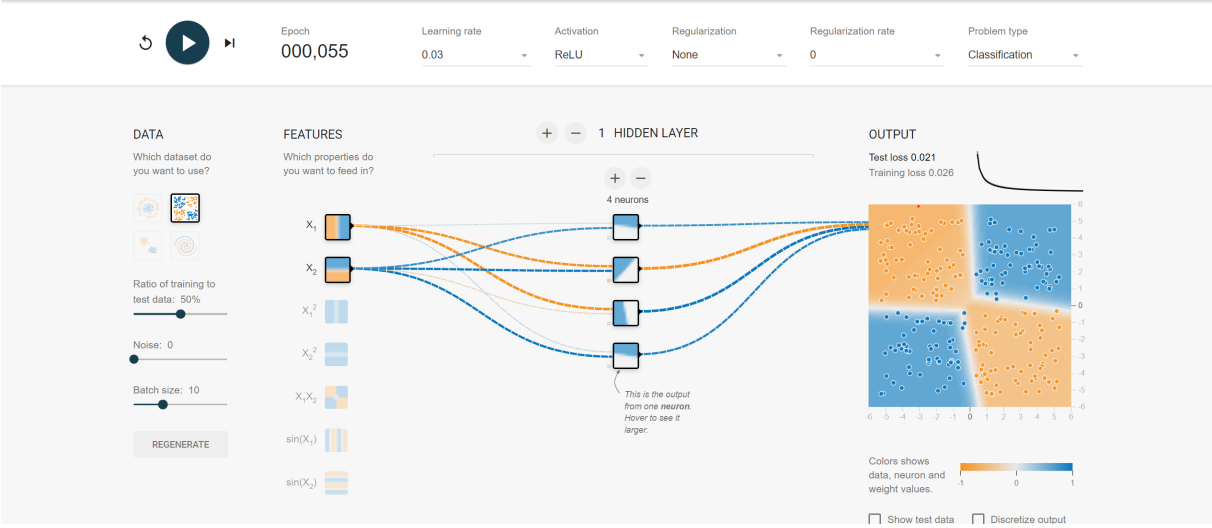


Figure A.3 Showing the created neural network which learned to fit the Exclusive or dataset.

| Activation Function | Epochs converged | Converged |
|---|---|---|
| ReLU | 54 | Yes |
| Tanh | 100 | Yes |
| Sigmoid | 200 | Yes |
| Linear | $\infty$ | No |

Table A.1 Results obtained for changing the Activation Function in the neural network shown in Figure A.1, using the Circle dataset.

| Activation Function | Epochs converged | Converged |
|---|---|---|
| ReLU | 55 | Yes |
| Tanh | 90 | Yes |
| Sigmoid | 250 | Yes |
| Linear | $\infty$ | No |

Table A.2 Results obtained for changing the Activation Function in the neural network shown in Figure A.1, using the Exclusive or dataset.

| Activation Function | Epochs converged | Converged |
|---|---|---|
| ReLU | $\infty$ | No |
| Tanh | $\infty$ | No |
| Sigmoid | $\infty$ | No |
| Linear | $\infty$ | No |

Table A.3 Results obtained for changing the Activation Function in the neural network shown in Figure A.1, using the Spiral dataset.

| Activation Function | Epochs converged | Converged |
|---|---|---|
| ReLU | 10 | Yes |
| Tanh | 15 | Yes |
| Sigmoid | 20 | Yes |
| Linear | 15 | Yes |

Table A.4 Results obtained for changing the Activation Function in the neural network shown in Figure A.1, using the Gaussian dataset.

# Appendix B Exploring Learning Rates

| Learning Rate | Epochs converged | Converged | Oscillating |
|---|---|---|---|
| 10 | $\infty$ | No | Yes |
| 3 | $\infty$ | No | Yes |
| 1 | 26 | Yes | No |
| 0.3 | 19 | Yes | No |
| 0.1 | 20 | Yes | No |
| 0.03 | 50 | Yes | No |
| 0.01 | 100 | Yes | No |
| 0.003 | 300 | Yes | No |
| 0.001 | 700 | Yes | No |
| 0.0001 | $\infty$ | No | No |
| 0.00001 | $\infty$ | No | No |

Table B.1 Results obtained for changing the Learning Rate in pre-set neural network architecture, using the Circle dataset and ReLU activation function.

# Appendix C Playing with Layers and Neurons

| Hidden Layers | Number of Neurons | Epochs converged | Converged |
|---|---|---|---|
| 1 | 1 | $\infty$ | No |
| 1 | 3 | 100 | Yes |
| 1 | 5 | 58 | Yes |
| 2 | 3 | 70 | Yes |
| 2 | 5 | 61 | Yes |
| 3 | 3 | 100 | Yes |
| 3 | 5 | 60 | Yes |

Table C.1 Results obtained for changing the number of hidden layers in pre-set neural network architecture, using the Circle dataset and ReLU activation function.

# Appendix D Optimisation Algorithms

```
// lets use an example fully-connected 2-layer ReLU net
var layer_defs = [];
layer_defs.push({type:'input', out_sx:24, out_sy:24, out_depth:1});
layer_defs.push({type:'fc', num_neurons:20, activation:'relu'});
layer_defs.push({type:'fc', num_neurons:20, activation:'relu'});
layer_defs.push({type:'softmax', num_classes:10});

// below fill out the trainer specs you wish to evaluate, and give them
   names for legend
var LR = 0.01; // learning rate
var BS = 8; // batch size
var L2 = 0.001; // L2 weight decay

nets = [];
trainer_defs = [];
trainer_defs.push({learning_rate:LR, method: 'sgd', momentum: 0.0,
   batch_size:BS, l2_decay:L2});
trainer_defs.push({learning_rate:LR, method: 'sgd', momentum: 0.91,
   batch_size:BS, l2_decay:L2});
trainer_defs.push({learning_rate:LR, method: 'adagrad', momentum: 0.0,
   batch_size:BS, l2_decay:L2});
trainer_defs.push({learning_rate:LR, method: 'adagrad', momentum: 0.91,
   batch_size:BS, l2_decay:L2});
trainer_defs.push({learning_rate:LR, method: 'windowgrad', momentum: 0.0,
   batch_size:BS, l2_decay:L2});
trainer_defs.push({learning_rate:LR, method: 'windowgrad', momentum: 0.91,
   batch_size:BS, l2_decay:L2});
trainer_defs.push({learning_rate:LR, method: 'adadelta', momentum: 0.0,
   batch_size:BS, l2_decay:L2});
trainer_defs.push({learning_rate:LR, method: 'adadelta', momentum: 0.91,
   batch_size:BS, l2_decay:L2});
trainer_defs.push({learning_rate:LR, method: 'nesterov', momentum: 0.0,
   batch_size:BS, l2_decay:L2});
trainer_defs.push({learning_rate:LR, method: 'nesterov', momentum: 0.91,
   batch_size:BS, l2_decay:L2});

// names for all trainers above
legend = ['sgd', 'sgd+momentum', 'adagrad', 'adagrad+momentum','windowgrad'
   ,'windowgrad+momentum', 'adadelta', 'adadelta+momentum', 'nesterov', '
   nesterov+momentum'];
```

Listing D.1 ConvNetJS code listing

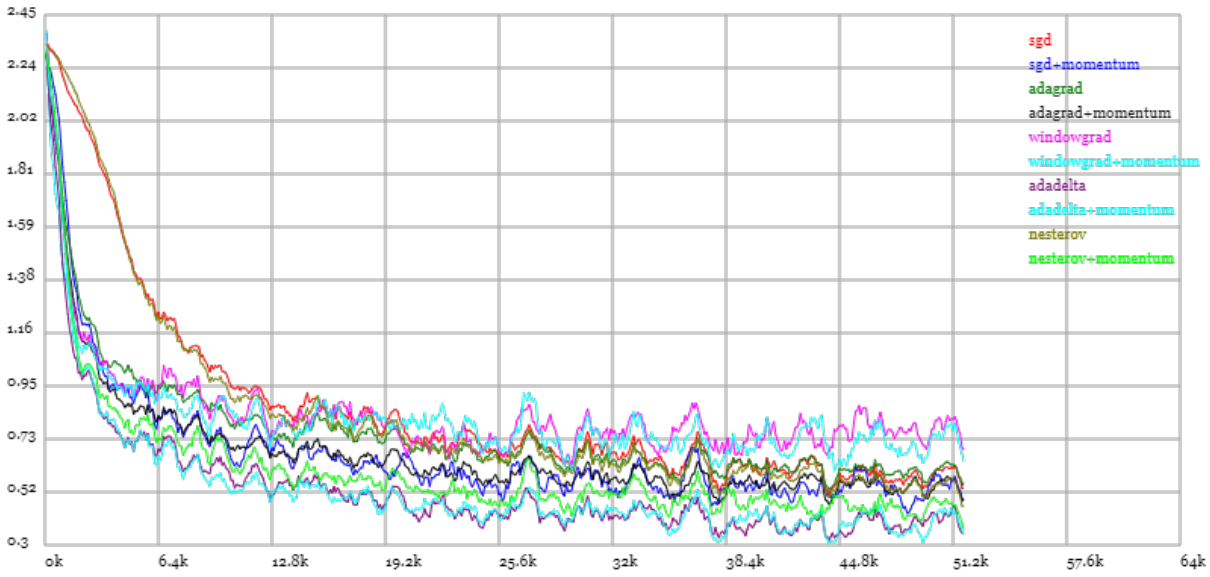


Figure D.1 Loss vs. Number of examples seen

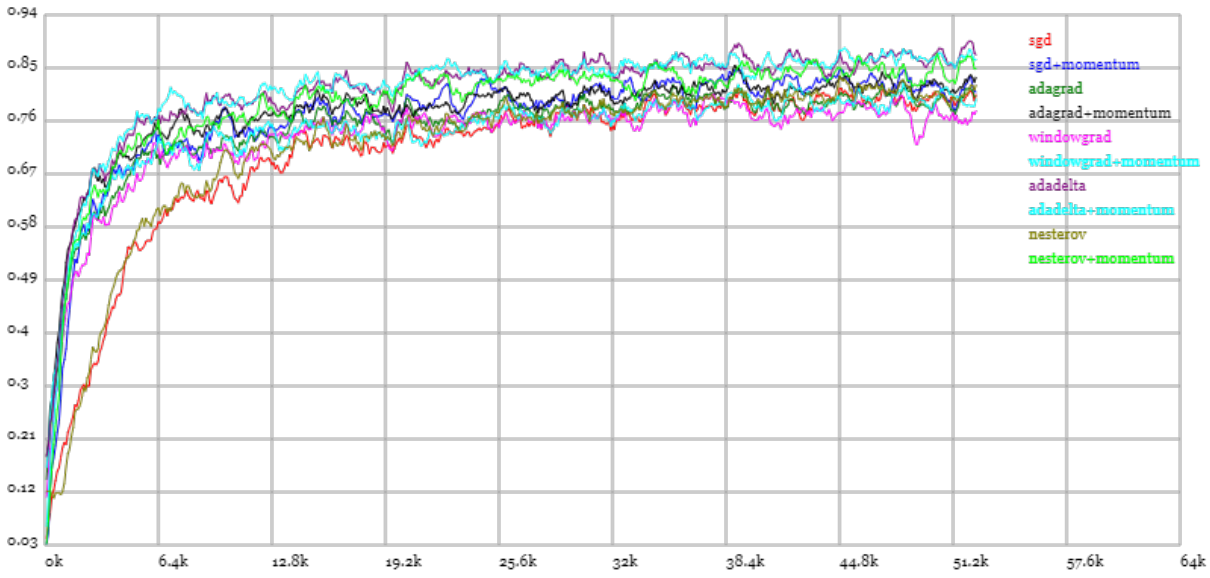


Figure D.2 Testing Accuracy vs. Number of examples seen

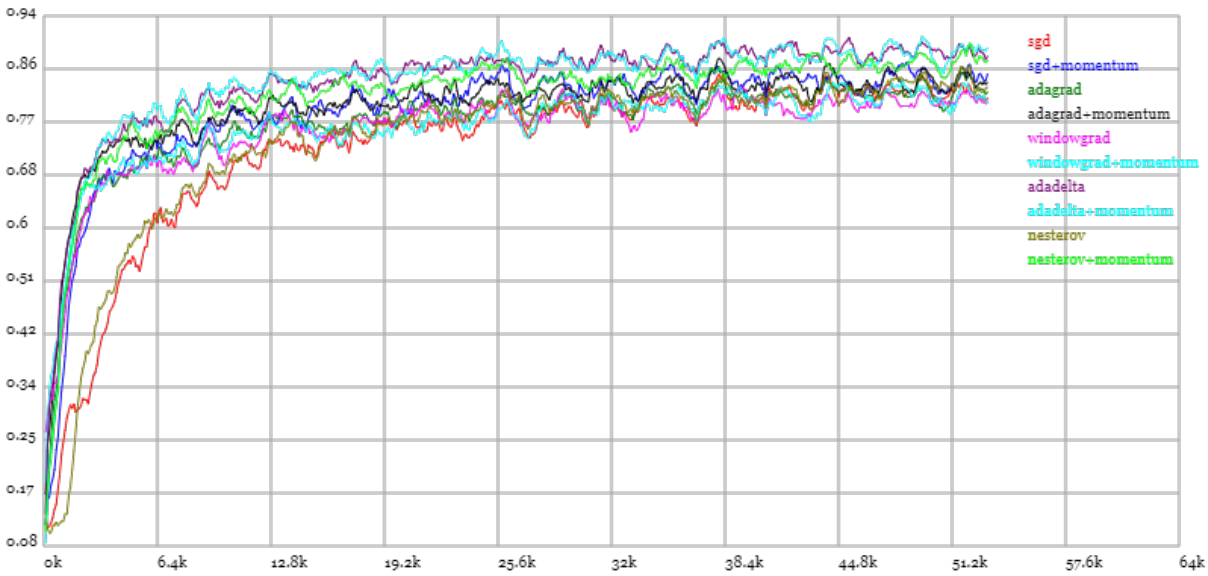


Figure D.3 Training Accuracy vs. Number of examples seen

# Plagiarism Form

## FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Declaration

Plagiarism is defined as "the unacknowledged use, as one's own work, of work of another person, whether or not such work has been published" (Regulations Governing Conduct at Examinations, 1997, Regulation 1 (viii), University of Malta).

I /~~We~~* , the undersigned, declare that the [assignment / Assigned Practical Task report / Final Year Project report] submitted is my /~~our~~* work, except where acknowledged and referenced.

I /~~We~~* understand that the penalties for making a false declaration may include, but are not limited to, loss of marks; cancellation of examination results; enforced suspension of studies; or expulsion from the degree programme.

Work submitted without this signed declaration will not be corrected, and will be given zero marks.

* Delete as appropriate.

(N.B. If the assignment is meant to be submitted anonymously, please sign this form and submit it to the Departmental Officer separately from the assignment).

| | |
|---|---|
| StudentName | Signature |
| StudentName | Signature |
| StudentName | Signature |
| Matthias Bartolo<br>StudentName |  <br>Signature |

| | |
|---|---|
| ARI3129<br>CourseCode | Advanced Computer Vision Individual Assignment<br>Title of work submitted |

17 October 2023
Date