Assignment 1 Report

Name: David Naddall

Student Email: [659586@student.inholland.nl](mailto:659586@student.inholland.nl)

# Part 2: A custom neural network

## Choices

### Loss Function

For this assignment, the Mean Absolute Error (MAE) loss function was chosen, MAE calculates the average absolute difference between the predicted values and the actual values. As a result, this function can provide a suitable, robust measure of error for the problem described in the assignment.

### Activation Function

The Rectified Linear Unit (ReLU) activation function was chosen for this assignment. ReLU is known for its simplicity and effectiveness, allowing the network to learn complex patterns while mitigating the vanishing gradient problem. Furthermore, ReLU is a common choice for regression problems such as this assignment.

### Hyperparameters

The Hyperparameters that were selected are:

* Learning Rate: 0.0001
* Number of Hidden Layers: 10
* Number of Epochs: 1000

The learning rate was chosen by iteratively checking where the loss would be the lowest. Meanwhile, with the number of hidden layers 10 was selected to add some complexity to the model. Lastly, the number of epochs was chosen to be 1000 to give the model enough passes to converge to a flattened amount of loss.

## Results

### Training

MAE: 1.490

The MAE on the training set indicates how well the model fits the training data. The final MAE value of 1.490 indicates the model's performance on the training data after 1000 epochs.

### Validation

MAE: 1.545

The MAE on the validation set provides insight into the model's performance on unseen data. A slightly higher MAE compared to the training set may indicate some degree of overfitting.

### Test

MAE: 1.493

The MAE on the test set serves as the final evaluation metric, providing an estimate of the model's performance in real-world scenarios.

Actual Weight vs Predicted weight (Custom neural network)

Actual Weight (g) Predicted Weight (g)

0 4900.0 4671.171739

1 4750.0 4035.223160

2 5750.0 4283.023273

3 3700.0 5723.124348

4 4300.0 4997.433260

.. ... ...

64 4725.0 3822.062473

65 3800.0 5323.481061

66 4250.0 4838.184753

67 6000.0 4576.063260

68 3200.0 5506.365171

Average actual Difference

1195.2115854379038

## Conclusion

### Analysis

With an average absolute difference of 1195.2115854379038 the model is quite off when it comes to predicting the weight of the pinguins using these inputs with a custom model. As previously said the validation loss being higher than the training loss can indicate a form of overfitting, However as the difference between the two is so low it may not be significant enough to consider the model overfitted.

### Discussion

Further analysis and possibly adjustments to the model architecture or regularization techniques may be necessary to improve its performance on unseen data. It's also important to consider other factors such as the quality of the input data, the representativeness of the dataset. Moreover, the amount of hidden layers can further increase the models complexity and perhaps yield better results. Using optimisation techniques such as adam can adjust the hyperparameters algorithmically to have optimal values for hyperparameters.

# Part 3: Using TensorFlow

## Choices

### Loss Function

For this assignment, just like with the custom neural network the Mean Absolute Error (MAE) loss function was chosen. this function can provide a suitable, robust measure of error as it is still a regression problem. Additionally, MAE can effectively mitigate the impact of outliers in the dataset.

### Activation Function

ReLU (Rectified Linear Unit) activation function was used for the hidden layer, while a linear activation function was used for the output layer. Using a linear activation function allows the network to directly output numerical values without any constraints making it easier to understand and use the results.

### Hyperparameters

* Learning Rate: Adam optimizer with a learning rate of 0.001 was selected.
* Number of Hidden Layers: 3 models were trained using different layers for each
* Number of Epochs: Training was performed for 1000 epochs.
* Batch Size: Batch size of 32 was used for training.

A smaller learning rate generally leads to more stable convergence but may require more epochs to reach optimal performance. The choice of one hidden layer with 16 neurons is a relatively simple architecture. For this task, where the input data may not be highly complex, a single hidden layer with a moderate number of neurons can capture the necessary patterns in the data without overfitting. For testing purposes 2 more models were trained with the same hyperparameters with the exception of the hidden layers. Model 2 had 2 hidden layers containing 64 and 32 neurons, respectively. Lastly, Model 3 had 3 hidden layers containing 128,64 and 32 neurons, respectively. A value of 1000 epochs was chosen to give the models enough iterations for the models to converge to a stable level of loss. Furthermore, a callback function was set to monitor the validation loss with a patience of 25. This ensures that the models do not train for an unnecessary amount of epochs. A batch size of 32 was chosen as this is a balanced number of batches.

## Results

### Training

Model 1:

MAE 0.3139

Epochs: The model stopped training at 165 epochs

Model 2:

MAE: 0.3038

Epochs: The model stopped training at 38 epochs

Model 3:

MAE: 0.2982

Epochs: The model stopped training at 38 epochs

### Validation

Model 1:

MAE: 0.3401

A graph with a line

Description automatically generated

Model 2:

MAE: 0.3433

A graph with a line

Description automatically generated

Model 3:

MAE: 0.3618

A graph with a line

Description automatically generated

### Test

Model 1:

MAE: 0.3618

Model 2:

MAE: 0.3618

Model 3:

MAE: 0.3685

Actual Weight vs Predicted weight (Model 1)

Actual Weight (g) Predicted Weight (g)

0 4900.0 4861.268066

1 4750.0 4902.505859

2 5750.0 5424.021973

3 3700.0 3366.647949

4 4300.0 4099.086914

.. ... ...

64 4725.0 5072.271973

65 3800.0 3472.678955

66 4250.0 3706.414062

67 6000.0 5356.810059

68 3200.0 3383.594971

Actual Weight vs Predicted weight (Model 2)

Actual Weight (g) Predicted Weight (g)

0 4900.0 4877.730957

1 4750.0 4922.409180

2 5750.0 5396.698242

3 3700.0 3395.897949

4 4300.0 4213.819824

.. ... ...

64 4725.0 5083.926758

65 3800.0 3453.418945

66 4250.0 3685.170898

67 6000.0 5321.006836

68 3200.0 3486.874023

Actual Weight vs Predicted weight (Model 3)

Actual Weight (g) Predicted Weight (g)

0 4900.0 4894.344238

1 4750.0 4933.943848

2 5750.0 5433.276855

3 3700.0 3375.501953

4 4300.0 4100.967773

.. ... ...

64 4725.0 5111.916016

65 3800.0 3460.728027

66 4250.0 3709.496094

67 6000.0 5363.407227

68 3200.0 3313.809082

Average actual difference:

Model 1 289.70206351902175

Model2 289.7065182008605

Model3 295.1013962013134

## Conclusion

### Analysis

Models 1 and 2 demonstrate similar average differences, indicating consistent performance in predicting the target variable across the dataset.

Model 3 exhibits a slightly higher average difference compared to Models 1 and 2. This suggests that Model 3 may have slightly lower accuracy or precision in its predictions.

### Discussion

The slightly higher average difference in Model 3 could be potentially influenced by the increased complexity introduced by the additional hidden layers and neurons. This could have implications of overfitting or increased variance, resulting in slightly less accurate predictions compared to Models 1 and 2. The training loss being lower than the validation loss can be an indicator of this.

# Comparison

## Loss Functions

Custom-made Model:

The custom-made model was trained using the Mean Absolute Error (MAE) loss function, which calculates the average absolute difference between the actual and predicted values. This loss function provides a robust measure of error and can mitigate the impact of outliers in the data. The Mean Squared Error(MSE) was also an option however, the MAE gives a representable number which makes it easier to analyse

Keras Models:

The Keras models were also trained using the MAE loss function, ensuring consistency in the evaluation metric across all models. This allows for a direct comparison of the performance based on the same criterion.

## Training Efficiency

Custom-made Models:

The custom-made model was trained using a manual implementation of gradient descent and backpropagation algorithm. While this approach offered flexibility and customization options, it also required more effort in tuning hyperparameters and optimizing training procedures.

Keras Models:

The Keras models benefit from the built-in optimization algorithms and training utilities provided by the Keras framework. This simplified the training process and ensured that the model was being trained correctly as it eliminated the possibility of errors being made in the making of the functions

## Generalization Ability

Custom-made Models:

The custom-made model was evaluated based on the performance on validation and test datasets, providing insights into the ability to generalize. The performance on unseen data indicated its capacity to capture underlying patterns in the data and make accurate predictions of the weight of the pinguins. Upon analysis with an average actual difference 1195.2115854379038, it was concluded that the custom made model was far off the mark when it came to predicting the weight of the pinguins.

Keras Models:

Similarly, the Keras models were evaluated based on their performance on validation and test datasets, allowing for a direct comparison of their generalization ability with the custom-made model. While compared to each other it was concluded that adding more hidden layers and neurons may have made the model to complex for this specific problem. The models got worse as the complexity grew. However, when compared against the custom-made model, the Keras models demonstrate a significant improvement in predicting the weight of penguins, with the best Keras model having an average actual difference of 289.70206351902175.

## Conclusion

In conclusion, both the custom-made and Keras models demonstrate their strengths and limitations in terms of performance, training efficiency, and generalization ability. While the custom-made model offers flexibility and customization options, the Keras models benefit from the convenience and efficiency of the Keras framework. Furthermore, minimizes the risk of having mistakes in the functions of gradient decent and backpropagation. The choice between these approaches depends on factors such as the specific requirements of the task, the available resources, and the desired level of control over the training process.