INTRODUCTION

The models described in this report were created with the aim to analyse data about human activity classification, with the aim to predict movements of people based on the signal of sensors they are wearing.

In the experiment there were 30 volunteers each performing 6 activities: walking, walking upstairs, walking downstairs, sitting, standing, and laying while wearing a smartphone on the waist. Through the phone embedded accelerometer and gyroscope, 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz have been captured.

The data set is composed of time series and for each recording it is provided:

- Triaxial acceleration from the accelerometer (total acceleration) and the estimated body acceleration.
- Triaxial Angular velocity from the gyroscope.
- A 561-feature vector with time and frequency domain variables.
- Its activity label.
- An identifier of the subject who carried out the experiment.

Different models have been created. Initially, models have been created utilising a subset of the dataset which include only body acceleration. The dataset has also been divided into training and validation. In this part, single layers neural network from scratch and the library Keras have been used. In the last part instead, the whole dataset and only the library Keras have been used.

ANALYSIS

For the first part, the dataset only contains body acceleration. A single layer neural network from scratch has been created using a multinomial logistic regression. A first model has been created using 10000 train steps, 10^{-3} learning rate, and SGD optimizer. A second model has been created utilizing an Adam optimizer. From Figure 1 it is possible to notice that the loss function in plot B reaches a lower minimum than plot A in less time, making the second model more efficient. Therefore, the Adam optimizer will be used to create also the following models.

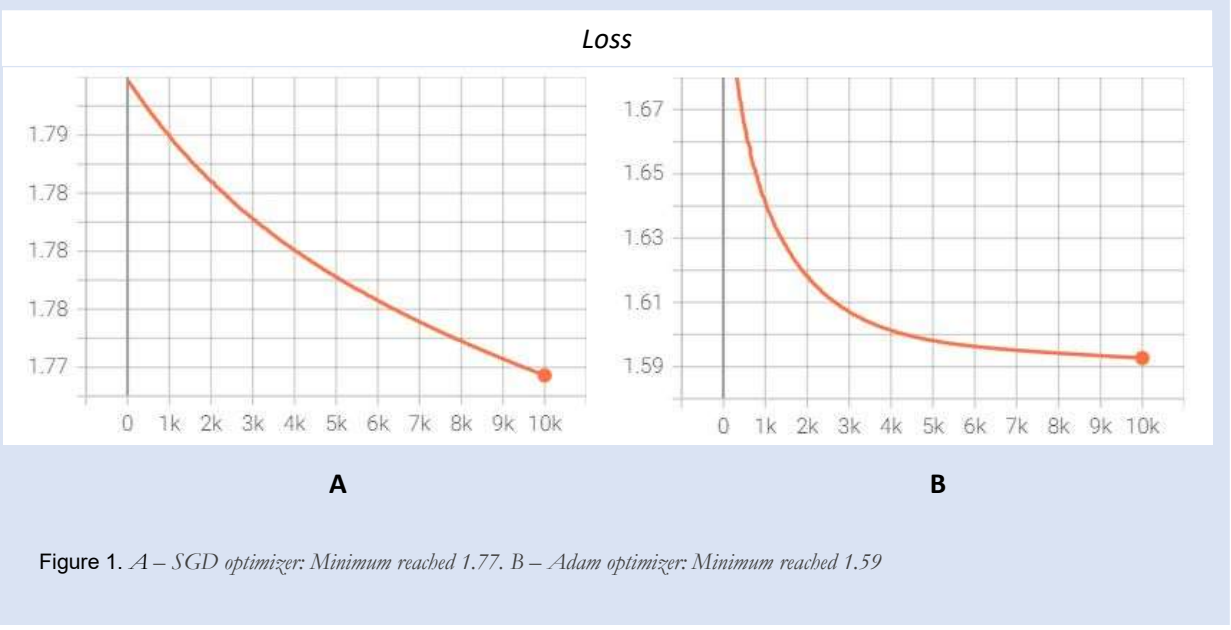

Figure 1. *A – SGD optimizer: Minimum reached 1.77. B – Adam optimizer: Minimum reached 1.59*

Subsequently, other models have been created with the aim to assess the best values of training steps and learning rate.

Figure 2. *A –Training Steps = 5000: Minimum reached 1.59. B –Training Steps = 15000: Minimum reached 1.58*

Comparing plot B of Figure 1 and the two plots of Figure 2, it is possible to notice that the minimum reached is similar for all three models, therefore, a training step of 5000 will be used because it makes the process faster and more efficient.

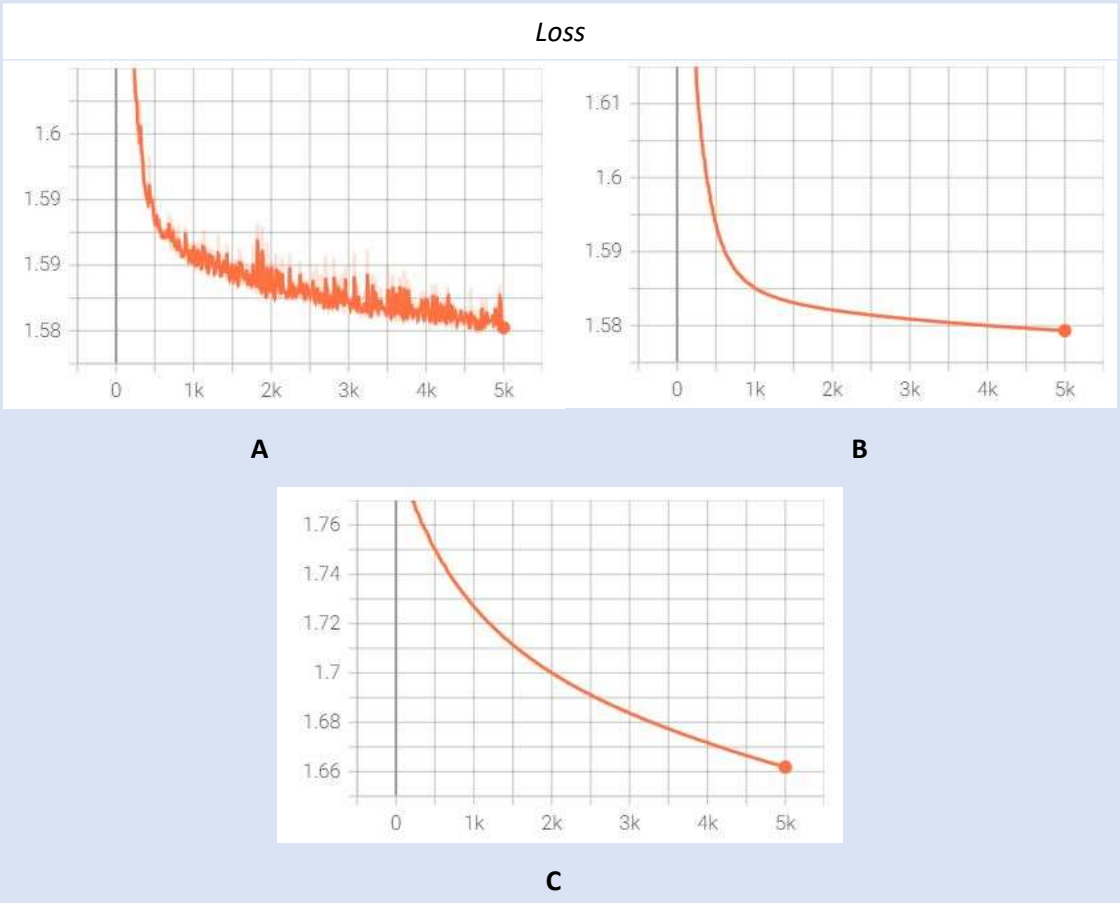

Figure 3. *A – Learning Rate = 0.1B – Learning Rate = 0.01. C – Learning Rate = 0.0001*

In Figure 3 are shown plots A, B, and C of the loss function having learning rate respectively of 0.1, 0.01 and 0.0001. It is evident that the learning rate is too fast in plot A and it is too slow in plot C. Plot B instead is similar to plot B of Figure 1 which has a learning rate of 0.001. However, comparing the plot of the behaviour of the accuracy calculated on the training dataset for these last two learning rates, as shown in Figure 4, it is clear that the value of 0.001 is the best choice.

Figure 4. *A – Learning Rate = 0.001. B – Learning Rate = 0.0001*

Accuracy, reported in Table 1, has been calculated for the training and validation datasets, using the chosen model which is characterised by 5000 training steps, Adam optimizer, and learning rate = 10^{-3}.

Dataset	*Accuracy*
Training	0.42
Validation	0.31

Table 1 *Values of accuracy calculated on the training and validation datasets.*

In the second part, the analysis is still done on a subset of the dataset which includes only body acceleration. In this section models will be created using convolutional neural network and the Keras API. Furthermore, as seen in the previous section, the optimizer used is Adam because it provides a more efficient model. The first model is composed of a 1-D convolutional layer, a Batch Normalization layer, a ReLu activation layer, a Global Average Pooling layer and a final dense layer to classify the human activity into the 6 classes. The values of *filters* and *kernel_size* are respectively equal to 36 and 4. This model is also characterised by 10 *epochs* and 32 *batch_size*. The accuracy for this model is around 80%.

Thereafter, tuning of hyperparameters has been performed starting with *epochs*. To take into consideration overfitting, a large value (100) for *epochs* has been chosen, and what has been actually tuned is the value of *patience* in the function *EarlyStopping*. This allows to stop the training when the loss function does not improve after a certain number of *epochs*, which is equal to the number of *patience.* This helps with overfitting issue. In Table 2 are reported the accuracy calculated on the validation set for all the models with different values of *patience.* It is clear that the best value for *patience* is 15.

Patience	Accuracy (%)
4	80
6	82
8	84
10	82
15	87
20	86

Table 2 *Values of accuracy for different values of patience.*

The second hyperparameter to be tuned is *batch_size* which represents the number of samples used to train the model. In this way small portion of the dataset are fed to the network, reducing the amount of memory used during the process and thus reducing time. Table 3 shows the accuracy calculated for all the models created with different number of *batch_size* and the value of 32 is the best choice.

Batch_size	Accuracy (%)
16	84
32	87
64	81
128	85

Table 3 *Values of accuracy for different values of batch_size.*

The third hyperparameter to be tuned is the *learning rate* that it determines the pace with which the model adapts itself with respect to the loss gradient. The new parameters of the model are calculated as follow:

$$\beta_{i-new} = \beta_{i-current} - \alpha\left(\frac{\partial CF}{\partial \beta_i}\right)$$

Where: β_{i-new} is the new value of the model parameter β_i; $\beta_{i-curren}$ is the current value of the parameter β_i; $\frac{\partial CF}{\partial \beta_i}$ is the derivative of the loss function with respect to β_i (the gradient), and α is the *learning rate.*

Learning Rate	Accuracy (%)
0.1	65
0.01	78
0.001	86
0.0001	81

Table 4 *Values of accuracy for different values of learning rate.*

Table 4 reports the values of accuracy calculated for the models having different *learning rates.* It seems evident that the best value would be 0.001. However, analysing the loss function of these models reported

in Figure 5, it seems that the best choice would be 0.0001 (plot D), because for the other values the rate appears to be too fast.

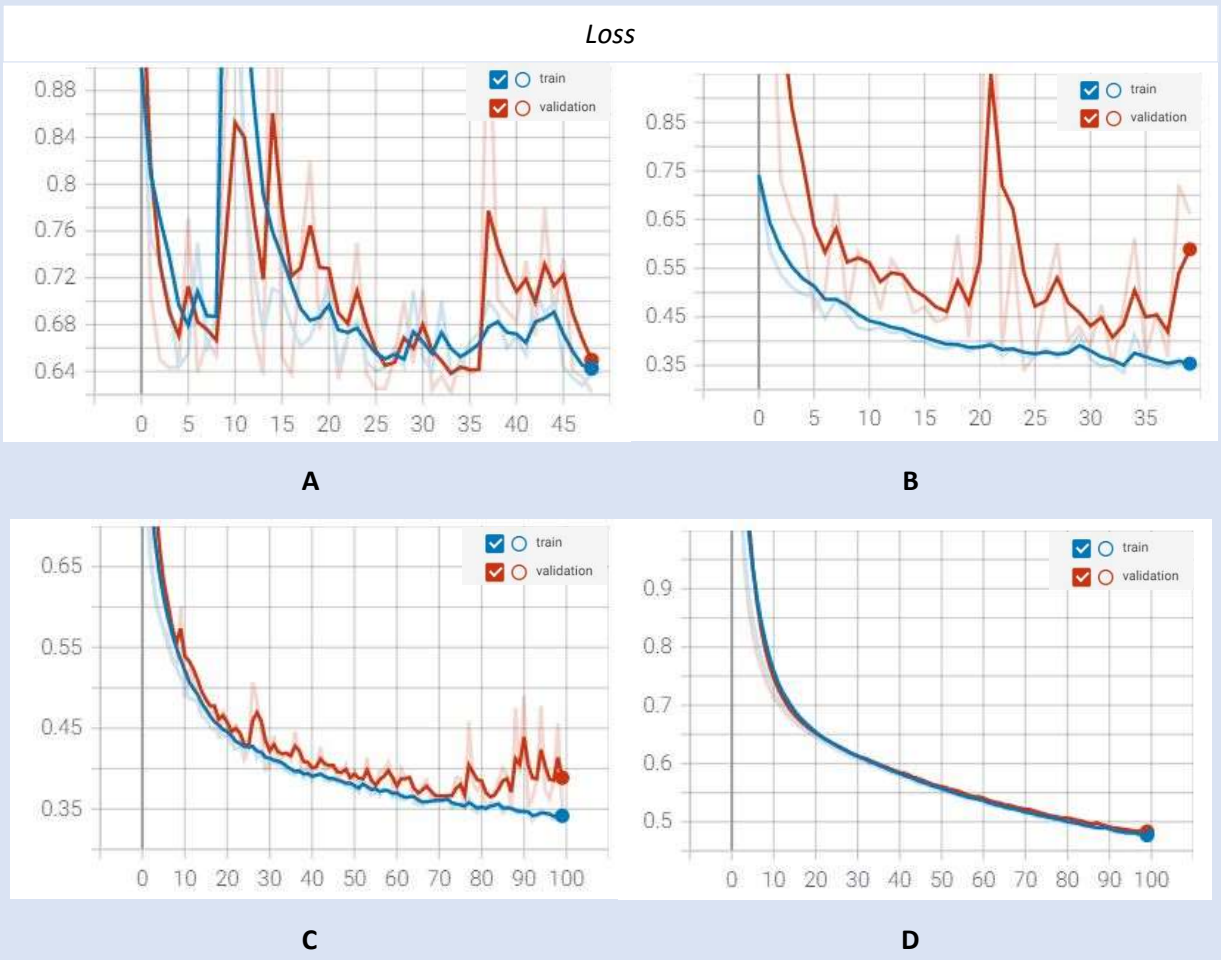

Figure 5. *A – Learning Rate = 0.1. B – Learning Rate = 0.01. C – Learning Rate = 0.001. D – Learning Rate = 0.0001*

The next step is to add more layers and tune the values of *Conv1D* function arguments: *filters* and *kernel_size*. The final model chosen is characterised by 100 *epochs* with *patience* set to 15, *batch_size* = 32, *learinig rate* = 10^{-4}, four convolutional layers with *filter* set to 64, 128, 256, 512 and *kernel size* = 8. The accuracy for this model calculated on the validation set is 90%

In the third part the analysis has been performed on the whole dataset which include the body acceleration, the total acceleration, and the Triaxial Angular velocity from the gyroscope. The first model created is identical to the final model chosen in part 2. Other models have been created to tune the *batch_size* but they did not bring any improvement to the model. The final model chosen has 100 *epochs* with *patience* set to 15, *batch size* = 32 and *learning rate* = 0.0001. Four convolutional layers with *Filters* equal to 64, 128, 256, and 512.

CONCLUSIONS

The analysis started applying neural network from scratch by using multinomial logistic regression for classification of human activity, and the dataset only considered body acceleration. It has been seen that using the Adam optimizer was the best choice because it made the model more efficient. 5000 training steps and 10^{-3} for the learning rate were enough. The accuracy calculated with this model on the validation dataset is 30% whereas calculated on the test set is 32%.

Subsequently, the Keras API has been used and the accuracy achieved after tuning all the parameters and changing the architecture of the model is 90% for the validation dataset, and 84% for the test set

Finally, it has been seen that this model used with the whole dataset reached and accuracy on validation dataset of 97% and 92% on the test set.

In conclusion, it has been noticed that the use of Keras API gives better results than the neural network from scratch and that the bigger the dataset the higher is the accuracy achieved.

These models can be improved by using a RNN component and better tune the hyperparameters but for time reason it will not be possible.