Maxime Fontana

The University of Sheffield

mfontana1@sheffield.ac.uk

As part of the module Speech Technology (COM4511), I present hereby the fulfilment of task 4 consisting in the implementation of a Voice Activity Detection (VAD) system.

1 Introduction

In this report we will present a solution to task 4 consisting in applying Deep Learning to perform binary classification on speech feature vectors as being either Silence or Voice. Firstly, we will introduce the methods surrounding the experimental environment. We will then analyse the diverse experiments that were carried out as part of this study.

2 Methods

In this section, we present firstly the type of algorithm and its different configurations taken into consideration in this study. We then introduce the related training framework we use to produce results.

2.1 Model

In this study, we implement a Feed-Forward Neural Network (FFNN), this is the most simple type of Neural Network that consists of fully-connected layers and in our case, 3 layers. Indeed, we set up a simple architecture with the following dimensions:

- Input Layer : $\mathbf{13}$
- 1st Hidden Layer : $\mathbf{250}$
- 2nd Hidden Layer : $\mathbf{100}$
- Output Layer : $\mathbf{1}$

To clarify this architecture, a set of additional mathematical operations has been utilised to facilitate training.
Firstly, we use Dropout, we therefore randomly, with a probability of 20%, assign a value of 0 to a neuron. That allows to prevent overfitting in order to generalise better. Secondly, we use the activation function ReLU in order to reduce non-linearity, this also facilitate training as it prevents from the vanishing gradient problem. This function is presented in the formula below:

$$ReLU(x) = max(0, x) \tag{1}$$

The aforementioned are used sequentially after each of the 2 hidden layer.
Finally, we use the provided Binary Cross Entropy with Logits Loss as a loss function, this apply the sigmoid function to our output neuron to perform our binary classification task. This function can be summarised as below:

$$S(x) = \frac{1}{1 + \exp{(-x)}} \tag{2}$$

2.2 Training

We now describe the training strategy adopted to optimise the model introduced in the previous section.

2.2.1 Data

In the study, we use the proposed split in the assignment description, we therefore do not make use of every samples in the entire dataset. Those files are gathered based on their respective prefix unique identifier and are then combined in respective Pytorch Dataset Class before being cast to a Dataloader.

2.2.2 Objective Function

As previously introduced, during training we aim at minimising Cross Entropy. This objective function is therefore taking as input our FFNN's sigmoid output and proceeds as follows:

$$BCE = -[tlog(p) + (1 - t)log(1 - p)] \tag{3}$$

2.2.3 Optimisation

Firstly, our model weight matrices are initialised randomly. Our model is then trained using *Mini-Batch Gradient Descent* with a learning rate initialised with the value **0.0001** and a batch size of **32**. As a further step to the optimisation process, our learning rate will be adjusted and decreased when a plateau is detected i.e. the loss has not significantly over a certain amount of weight updates. Cross validation was decided not to be implemented as the amount of data available for training was significant enough and therefore representative enough not to shift the validation and training sets.
Furthermore, a Early Stopping criterion is implemented, the source code is not to be taken credit for even though the adaptation is, the original source code is referenced in the class dedicated to that algorithm, this criterion checks

after each epoch that the validation loss is descending too, if it has not after 2 epochs, then it stops the training process, saves the checkpoint for which the validation loss was the lowest and save the model parameters accordingly.

2.2.4 Performance

We hereby report the performance and the objective function values acquired over the training process. The loss was acquired over every batch of size 32 and was then averaged per epoch.

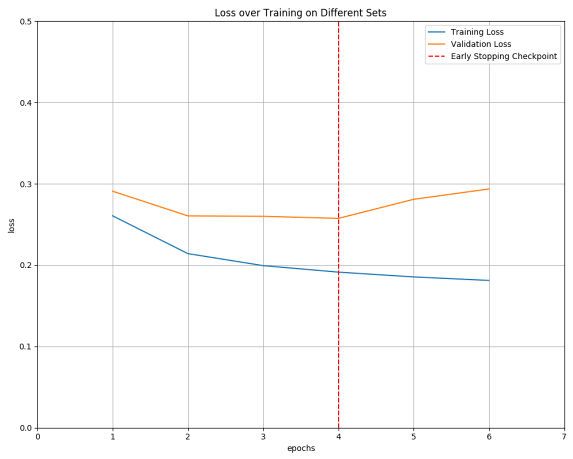

Figure 1: Loss during Training

On the figure above, we can see that the loss is decreasing over training, the vertical line shows the checkpoint saved by the Early Stopping method. This checkpoint is saved when the validation loss starts increasing, which demonstrates a state of overfitting, this therefore why the training loss will keep on decreasing until a possible plateau and the validation loss however, will start increasing again demonstrating that the current model starts to generalise poorly. Subsequently, this condition is assessed for 2 consecutive epochs to make sure the model's progress was not stuck in a local optima, if no improvement noticed, it terminates training.
The final objective function values recorded on the Training, Validation and Testing sets are summarised in the table below.

	Training	Validation	Test
BCE	0.18	0.26	0.34

3 Evaluation

We now present the evaluation performed after training and optimisation of our FFNN. We describe the different metrics acquired on our classification, firstly Accuracy, secondly the Detection Error Trade-off (DET) and lastly, the Equal Error Rate (EER).

3.1 Methods

3.1.1 Accuracy

Accuracy represents the fraction of true predictions our model managed to get right. It can be calculated as shown below :

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

When making a random prediction, the classifier will have a higher chance to get a specific class right, in our case 'Silence' than the other. This is due to the fact that our dataset splits are imbalanced, indeed :

- Test Set : $\mathbf{21\%}$ of 'Voice' samples
- Validation Set : $\mathbf{18.6\%}$ of 'Voice' samples

This bias is still to be taken into consideration when the model is trained, however, we illustrate our results with our random/trained weights in the table below:

	Test	Validation
Size	1.6 M	2.37 M
Distribution	21% / 79%	19% / 81%
Accuracy Random (0)	78%	81%
Accuracy Random (1)	21%	18%
Accuracy Trained	85%	90%

We therefore aim at improving our evaluation process by trying to acquire more meaningful binary classification metric results to give a clearer understanding of the model.

3.1.2 Detection Error Trade-Off

This metric represents the false rejection rate against the false acceptance rate. It is a useful metric to use in a classification context. Using our trained model we can plot this metric : We can see in Figure 2 above that the validation set False Rejection rate decreases faster than the test, this is due to the fact that the validation is directly used for optimisation and therefore the model adapts a lot faster and in a more accurate way. This curve shows that the false negative decreases, which means implicitly that true predictions are increasing. However, the 'False Alarme' rate i.e to classify a silent sample as voice increases in both cases. That is a poor attribute to a binary classification algorithm. However, this curve does not depict a full picture of the model's capacity as the mis-classification cost is not estimated, one should therefore ass whether a false negative is as important as a false positive or not.

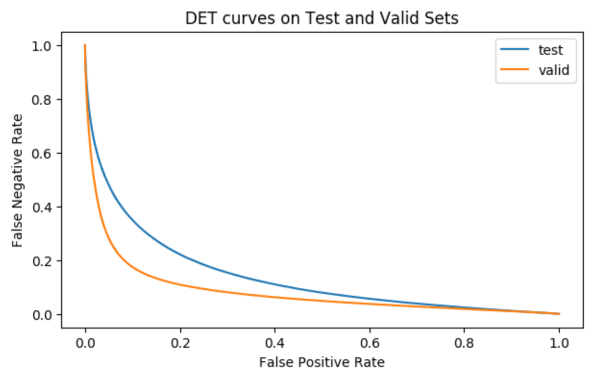

Figure 2: Detection Error Trade-Off Curve

3.1.3 Equal Error Rate

The ERR can be calculated from the receiver operating characteristic (ROC) curve, it is the point on the curve where the false positive rate and false negative rate are equal, it can be obtained by computing the intersection between the ROC curve and the linear function $x = 1 - y$. It can also be calculated from the DET curve but using this time, the linear function $x = y$. The smaller the EER value, the better the quality of the classification as the false positive rate would be smaller and the true positive rate higher. The EER provides a nice global view of the attributes of a classification, however it still does not fully assess the importance of a mis-classification compared to the other, for example a model with more mis-classification of voice would be better than a model with more mis-classification of silence. Which does not fully reflect the reality, we would rather spend time on overkill computations rather than miss a key speech signal.
The 2 figures below summarise the results obtained on the validation and test set respectively.
We can observe, as before with the DET curves, that the

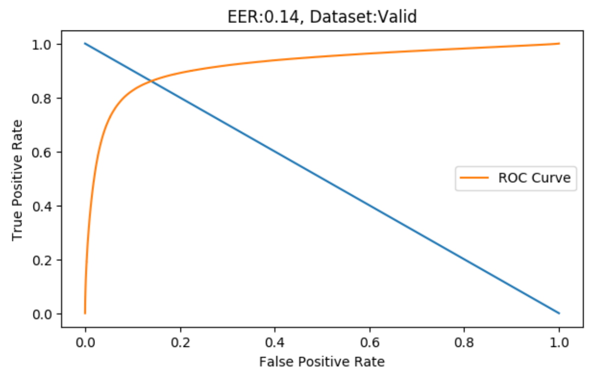

Figure 3: EER Validation Set

validation exhibits better results than the test set. This is again due to the fact that the model was implicitly built to perform well on this partition. Overall, the EER achieved

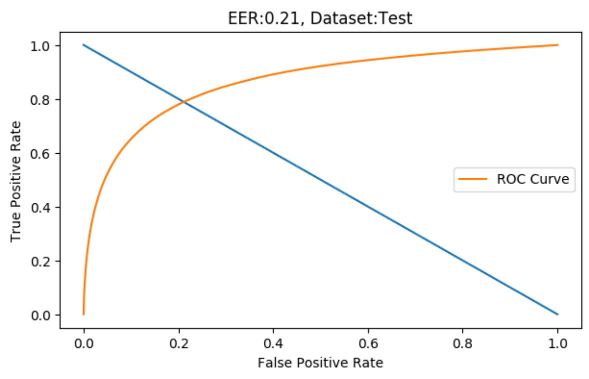

Figure 4: EER Test Set

on the test dataset is good and highlights and a decent classification efficiency.

4 Replication of Results

Please refer to the README.md file given in the provided zip file to replicate the presented results and get to know the code architecture. Please, make sure the audio and labels folder are present in the code folder. *Please Note:* Parts of this code has been directly imported from the following source : https://github.com/Bjarten/early-stopping-pytorch. This was **1.** To implement EarlyStopping in *PyTorch* as a class, and **2.** To render the plot showing the saved checkpoint by this method.

5 Conclusion

In this study, we have presented the model put into practice for the task, we have introduced how we optimised it in a binary classification task configuration. We have then carried out a comprehensive assessment using diverse metrics. Finally, we have shown how to replicate the obtained and further test the current solution in a practical way.