UNIVERSITY OF ATHENS
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

Deep Learning for NLP

Student name: *Lampropoulos Konstantinos*
sdi: sdi1800092

Course: *Artificial Intelligence II (M138, M226, M262, M325)*
Semester: *Fall Semester 2023*

Contents

1. Abstract

In this task,we were asked to develop and tune an Neural Network with stacked Bidirectional RNNs(which from now on for simplicity will be called RNN).RNNs are used to predict the next words in a sequence,which can be really helpful in an NLP task,since we can predict a whole sentence from just previous words.To tacke this task,we will first pre-process out dataset,by removing stopwords,removing whitespaces etc.Then we will convert our text to a list of indices ,where each indice represents the location of the word's vector in the embedding matrix.Finally using the optuna framework and some old-fashioned tuning and experimenting,we will get the best model and use it to predict the labels for the test set.

2. Data processing and analysis

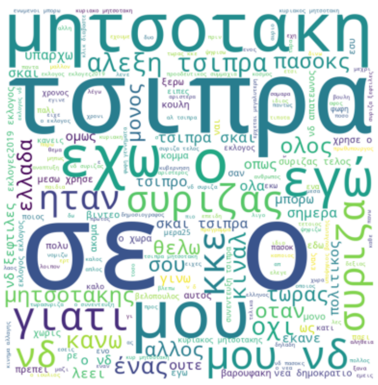
2.1. Pre-processing

In this section we will cover the pre-processing techniques involved.To begin with,we drop the unnecessary columns(NewID , Party) since we will not be using them in the classification process.After that, we convert all the text column in lowercase letters,remove the accents ,as well as, the stopwords and replace them with a single whitespace character.Following that,we replace multiple whitespace characters with a single white space,remove any links and non-greek words as well as retweets (and replace them with single whitespace characters).Finally,we lemmatize the text column to replace words with their dictionary values.The purpose of this pre-processing action is to breakdown the text and remove any unnecessary words and also provide the classifier with the dictionary value of the words so they are more easily recognized.

2.2. Analysis

2.2.1. Word Cloud.

In a word cloud,the most common words have greater size compared to the rest.Below I showcase an image with the results of the wordcloud implementation:



As we can see ,words like: Μητσοτάκη,Τσίπρα,νδ,συριζα, are really common in the database.We also have some words like: ο,σε,εχω ,which shouldn't be kept after removing stopwords.These should be removed for better results,but were not in this project.

2.2.2. *Token Frequency.*

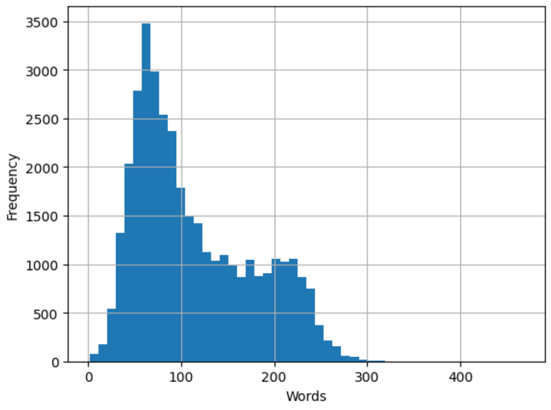
Token frequency is used to find the most common words in the dataframe.In this dataframe our 10 most common words are:

word	count
ο	18181
τσιπρα	9889
μου	8189
νδ	8168
μητσοτακη	6883
σε	4528
εκλογος	4298
εχω	4207
συριζα	2943
συριζας	2254

We can see that some stopwords appear as most common words.These should be removed manually as they may impact the training of the model,but were not for this project.As we can see some of the most common words are τσιπρα,νδ,μητσοτακη which means that most of the common words are correctly identified and will likely help us better identify the sentiment

2.2.3. *Text Length Distribution.*

Text length distribution is a method which finds the most common text lengths in the dataframe.



As we can see,the most common text lengths are between 40 to 80 words,which is the size of a common text or tweet.

2.3. Data partitioning for train, test and validation

The dataset remained the same as the one given in the assigment.

2.4. Vectorization

For the vectorization,the Word2Vec model was used with these parameters: vector size 300,window 5,minimum count 2 and negative 10. Vector size means that each word will be represented by a vector of 300 values,window 5 means that the model will check up to 5 words before and up to 5 words after,to learn the context of the current word.Minimum count 2 means that words appearing less than 2 times will be excluded,and negative is used to speed up training as it adjusts only a number of weights for each sample.Below is an example of the most similar words of a certain word:

word	Similarity Rate
Μητσοτακης	0.5311290621757507
συρρικνωσει	0.3812375068664551
φορων	0.3534460961818695
μεγαλοεπιχειρηματιος	0.3458263874053955
προχωρησει	0.3422377109527588
μητσοτακη	0.339499831199646
λουτσεσκος	0.33809322118759155
πειστικος	0.321236252784729
περικοπω	0.3175898790359497
διαλογο	0.31199774146080017

As we can see the most similar words to μητσοτακη ,are words that have some context to the nature of the current word(μεγαλοεπιχειρηματιος,πειστικος), or some similarity(Μητσοτακης,μητσοτακη).

It is somewhat interesting that the word Μητσοτακης,has a capital letter,since all words were converted to lowercase.This may be due to different encoding to the first letter of the word(since accents were removed correctly).

2.5. Embedding Matrix and sentence conversion to timesteps

The embedding matrix is a matrix where the vectors of each word in the dictionary of the word2vec model are stored based on their keys.For example if the key of the word 'μητσοτακης' is 5,the correspoding matrix will be stored at the 5-fth place of the matrix. The last step of the date pre-processing is to convert the senteces to sequence of indexes.In order to do that we check for each word in each sentence,if the word exists in the dictionary of words and their corresponding keys(created by the word2vec model) and if it does,we replace the word with the key. After that,we convert all the sentences to the length of the maximum length of a sentence found across all train,validation,test sets and lastsly we create our dataloaders.

3. Algorithms and Experiments

3.1. Experiments

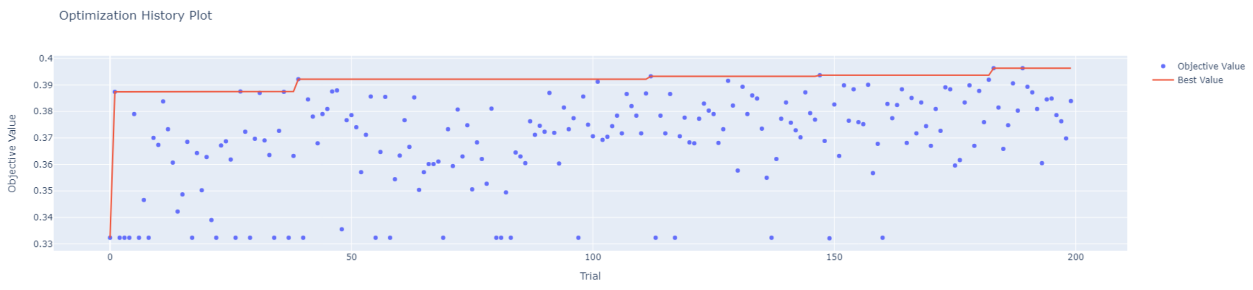
For the experimenting ,the Optuna framework was used.The optuna framework creates a study consisting of all the trials performed.Furthermore,the optuna framework requires an Objective Function to be declared and whether this function should be maximized or minimized.In this assignment,the objective function was to be maximized,as I chose to use improvement of accuracy and not minimizing of loss.In this particular dataset,f1 and accuracy are both useful metrics and valid,since the dataset is balanced.The hyper-parameters specified in the optuna framework along with their potential values are listed on the array below:

HyperParameter	1st Value	2nd Value	3rd Value	4th Value
Hidden Layer Size	16	32	64	128
Number of Hidden Layers	1	2	3	
Cell Type	LSTM	GRU		
Gradient Clipping	Yes	No		
Dropout Probability	From 0.0	To 0.5		

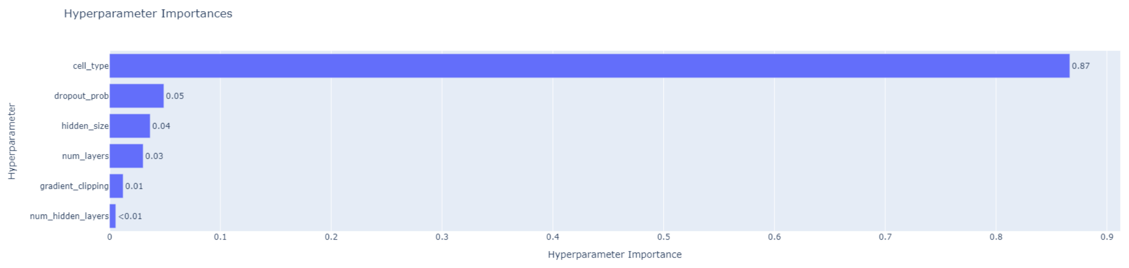
For the optimizer and loss function,Adam and CrossEntropyLoss were used,as were specified at the assignment.Furthermore, the learning rate was set to 1e-3 and Epochs were 1 for the whole Optuna training process.Both of these parameters were configured after.

3.2. Hyper-parameter tuning

Firstly we can check the optimization history:

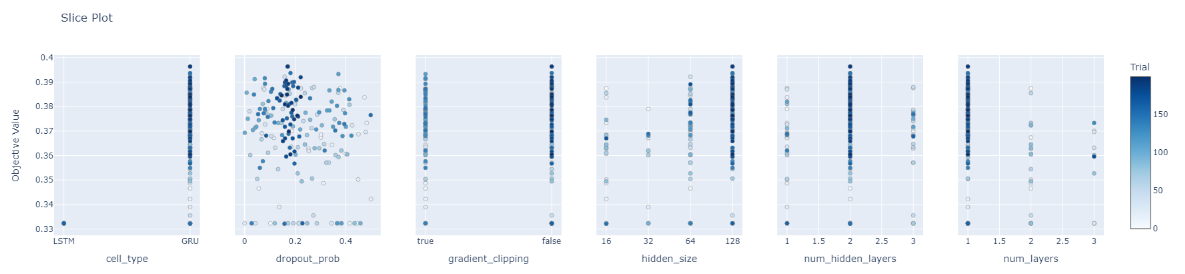


This is of little importance,but we can see that the best model was found at almost the end of trials.But we could have found a very similar(performace wise) model very early,at almost 50 trials.Next we will see the importance of each parameter to the performance of the model.

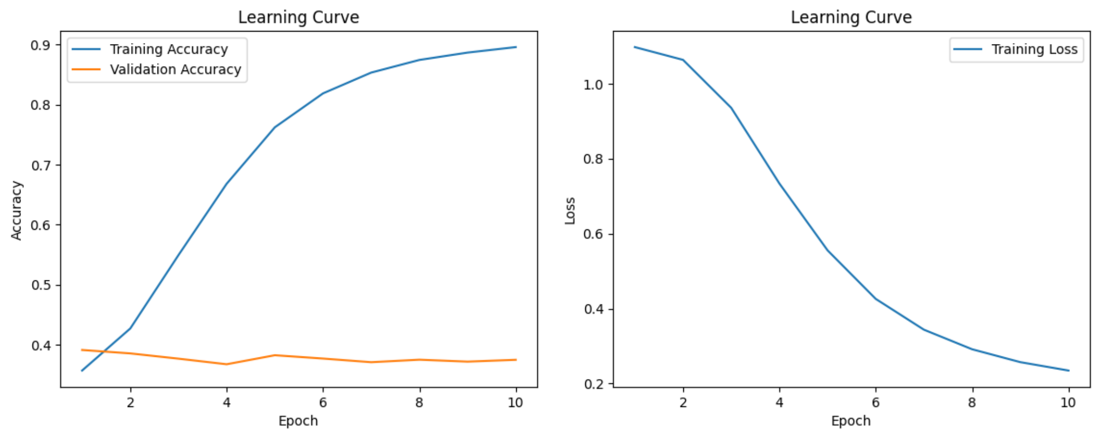


It is obvious that the parameter with the highest importance in the performance of our model is the type of cell used(LSTM or GRU).This was expected,since even though

the size or the number of hidden layers are important(as well as the other parameters) ,the type of cell is the foundation of the Neural Network and will be extremely impactful in its performance.The rest of the parameters showcase to be of similar importance,with gradient clipping and the number of hidden layers to be of least importance.The fact that dropout was significantly more important than the number of hidden layers was interesting to me,and the deduction i could come to,was that even with 1 hidden layet the model wasn't underfitting,so increasing the number of layers did little to improve the accuracy on the validation set.
Lastly we will look at the performance of each value of each parameter:

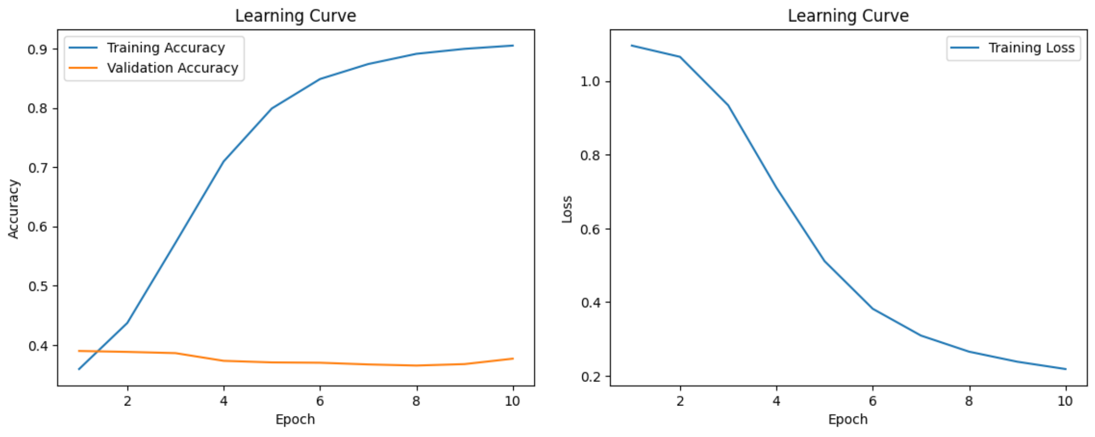


As we can see GRU was used in almost all the testing and yielded much better results,gradient clipping returned promising results for both values,but when not used there was a little increase in the performance.Furthermore,the size of hidden layers 128 was preferred with 2 hidden layers and as for the number of stacked Reccurent Neural Networks,1 was chosen.Finally we can see that droupout values of 0.0 to 0.2 were the best,so dropout was needed.This may be the reason GRU was preferred over LSTM,since dropout cannot be performed on LSTM cells.
After the tuining using the optuna framework,I performed some tuning manually(mainly concerning the number of epochs, learning rate and the overfitting prevention). To begin with,I experimented with the number of epochs,which I set to 10.Below are the learning curves:

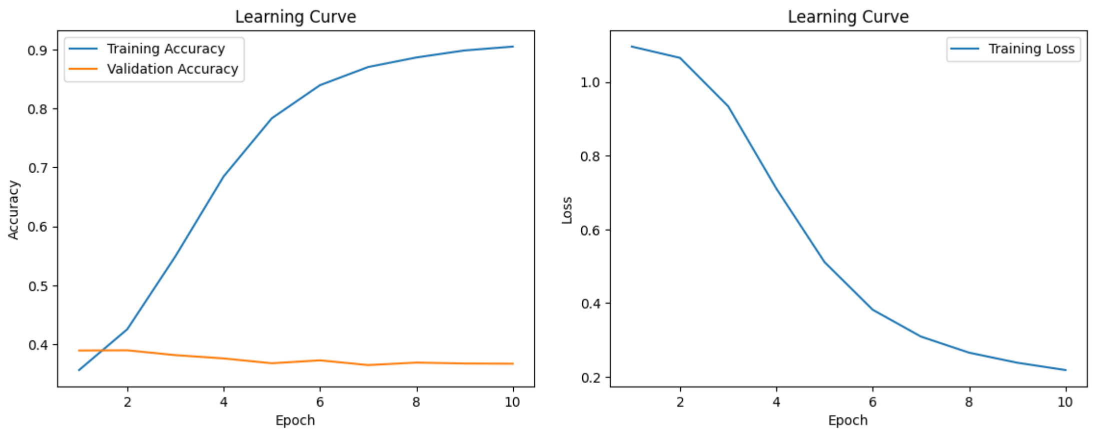


We can see major signs of overfitting which occurs even after the 2nd epoch.Furthermore we can see that even though the accuracy of then training set dramatically increases with each epoch,the validation accuracy remains roughly the same.We can see though that at the first epoch both scores are relatively the same and so,we can deduct that our model doesn't benefit from the increase of epochs,and 1 singular epoch is enough.

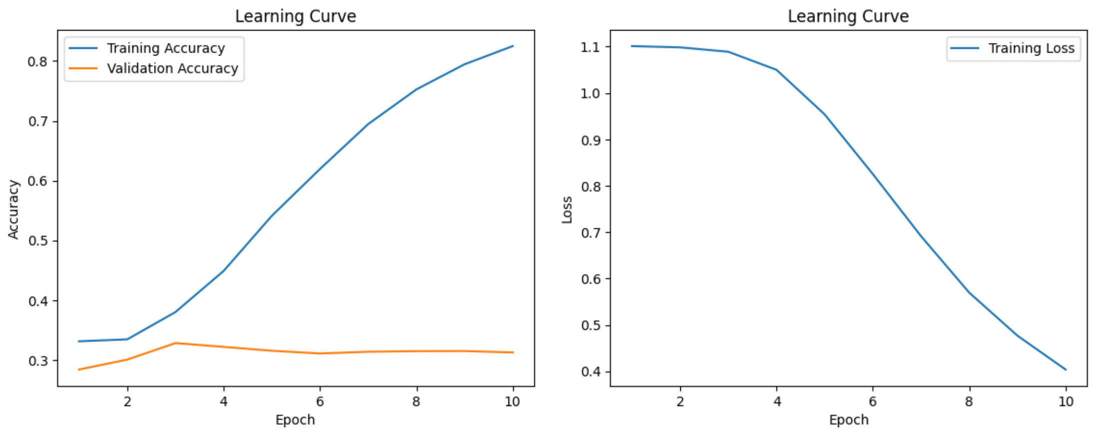
All these tests were done using batch size of 128,but what if we increase/decrease the batch size? Will epochs then help our model and incrase its accuracy?
Firstly we experimented with lower batch sizes,starting with 32:



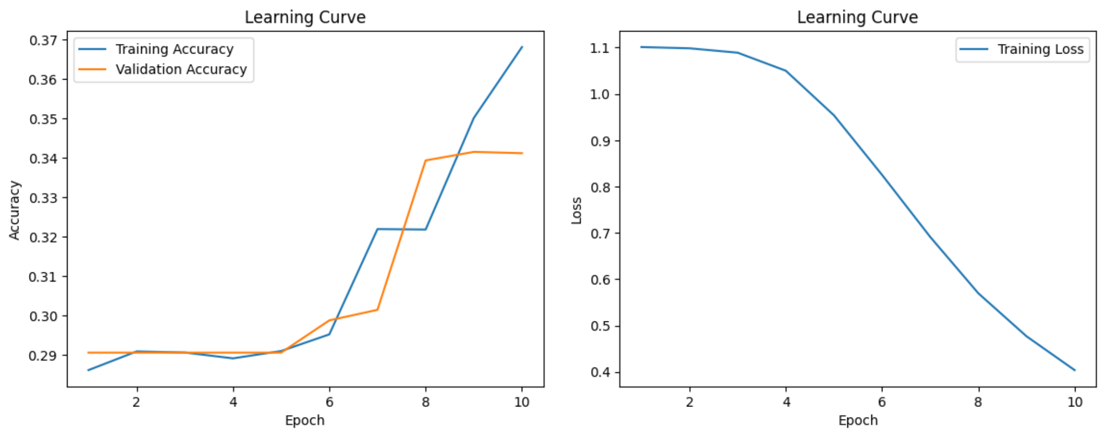
and 64:



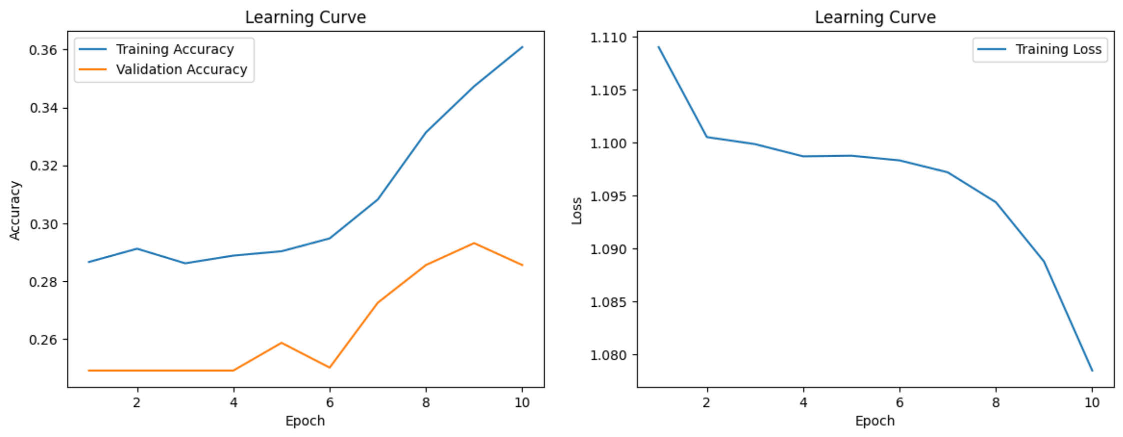
We can see that lower batch sizes do not help our model generalize through the course of epochs,and validation accuracy remains the same. By further increasing the batch size(128 is represented by the first experiment with epochs) to 1024 we get:



Still the signs of overfitting are immense,and the validation accuracy is really low.Lets see with a really high batch size what happens:



The above learning curves are with batch size 6000.We can see that with such a high batch size,our model starts to generalize better through the epochs, but still the accuracy remains significantly lower compared to our best one (39.6%).Lastly I experimented with a batch size of 7000:



With this batch size,our model again overfits and the accuracy score on the validation set is really low.Thus we can deduct that the number of epochs does not help our current model,and the best number of epochs is 1.

4. Results and Overall Analysis

4.1. Results Analysis

These results are expected.NLP is a difficult task and since the dataset is in greek it is even more difficult,since most support for NLP is for english,Having that in mind,and the fact that some of the items in the train dataset are falsely classified by default,we can reach to the conclusion that the accuracy for this model is acceptable.If the train dataset was correctly classified,we could maybe reach accuracy of 50% and more.The metric used for the tuning of the model was the Accuracy metric which compares the number of correctly classified items to the total number of items in the dataset.This metric is not always an acceptable metric and other metrics are preferred (F1-Score).In our case though,where the dataset classes(Negative,Neutral,Positive) are balanced ,it can be used as an valid metric.

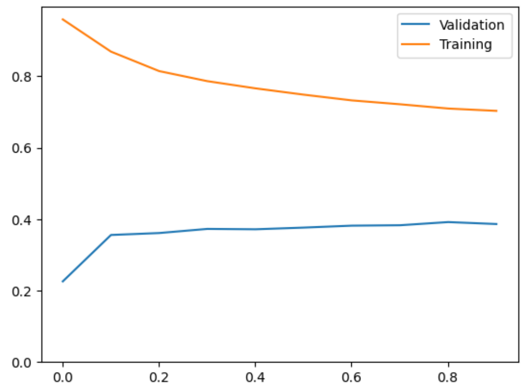
4.1.1. Best trial.

The best trial returned an accuracy score of : 39.6% with parameters:

- Number of Stacked RNNs: 1,
- Type of Cell: GRU,
- Number of Hidden Layers: 2,
- Size of Hidden Layer: 128,
- Gradient Clipping: False,
- Dropout Probability: 0.17125906907337562

4.2. Comparison with the first project

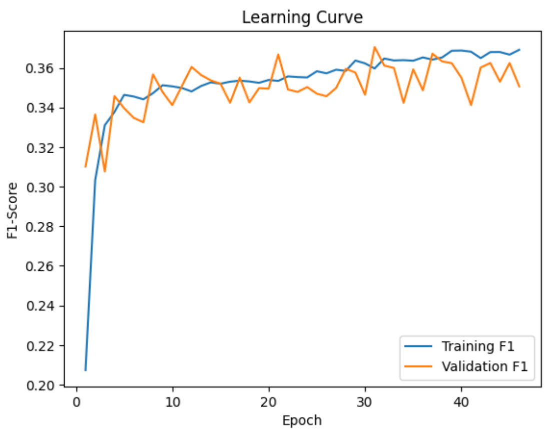
Compared to the fist project,our scored increased and we managed to prevent overfitting to a much greater degree.Below is the learning curve when using the logistic regression classifier:



This learning curve shows the f-score for training and validation.There is a huge difference between both f-scores which would in time decrease and no overfitting would occur,but that would need a much much bigger dataset.So even though perfomarce remained relatively the same,we managed to make our model generalize better to new unseen data.The best f1-score for this is 37.2%.Since our dataset is balanced both f1 and accuracy will converge to the same value,so we can assume that accuracy score will be near 37.0% too.

4.3. Comparison with the second project

Compared to the second project,our scored increased,but both NNs prevented overfitting.Below is the learning curve when using the FeedForward Neural Network:



The accuracy from the training of this model was : 37.2%,which is a bit lower than our current accuracy.Still boths NNs could benefit from a larger dataset and no falsely classified items in the training set.

5. Bibliography

References

[1] Pytorch And Neural Network Labs found on e-class.