Τεχνητή Νοημοσύνη 2

Εργασία 2

Στέφανος Μπακλαβάς 1115201700093

For the purposes of this hw only exam 3 was completed.

3.Model

3.1 Data Pre-Proccessing

The data pre-proccessing is similar with that of the hw1.First we remove every whitespace bigger than two spaces .Then we tun capital letters into small letters. An extra step was done by removing all numbers and emojis of the tweets but then it was found that our model performs better if we leave them and replace the with the mean vector of the tweet that they come from.

3.2 Glove Vectors

The glove vectors set that is used is the ‘glove.twitter.27B.50d.txt’.This set was chosen because it is trained on twitter data which is also the theme of our model. Each glove vector was put into a dictionary with the word as a key and the vector as an item. Then for each tweet of our dataset we found the words of the tweet that were dictionary keys and represented each tweet as the mean vector of its word vectors. If a word did not existed into the dictionary then we replace it with the mean vector of the tweet from which it comes from.

3.3 Model

There were done a lot of tests as it seems into the array below. The optimizer that was used in all of them was the torch.optim.SGD .In our dataloader we used batch_size = 64 for training and 32 for validation because validation set is a lot smaller. The symbols h1,h2,h3 correspond to the hidden layers of the model and the scores are the max scores of each model performance. Recall score was not used as it had the same result as precision.

*CEL = CrossEntropyLoss

**In example 10 we also put a softmax function after output layer in our model. This does not happen in the other examples because CEL passes the input through a softmax function.

Test#	Loss function	Activation function	Learning rate	h1	h2	h3	f1_score	Precision score
1	CEL	ReLU	1e-2	128	64	---	0.528739	0.578682
2	CEL	ReLU	1e-1	128	64	---	0.539164	0.586694
3	CEL	ReLU	1e-3	128	64	---	0.558783	0.522369
4	CEL	ReLU	1e-1	128	64	32	0.551531	0.606226
5	CEL	ReLU	1e-2	128	64	32	0.515286	0.571701
6	CEL	ReLU	1e-2	64	32	16	0.529495	0.585053
7	CEL	ReLU	1e-1	64	32	16	0.546069	0.595038
8	CEL	ReLU	1e-1	256	128	64	0.620764	0.667308
9	CEL	ReLU	1e-2	256	128	64	0.552178	0.591030
10	NLLLoss	ReLU	1e-1	256	128	64	0.620764	0.667308
11	CEL	LogSoftmax	1e-1	256	128	64	0.356515	0.479991

3.4 Results

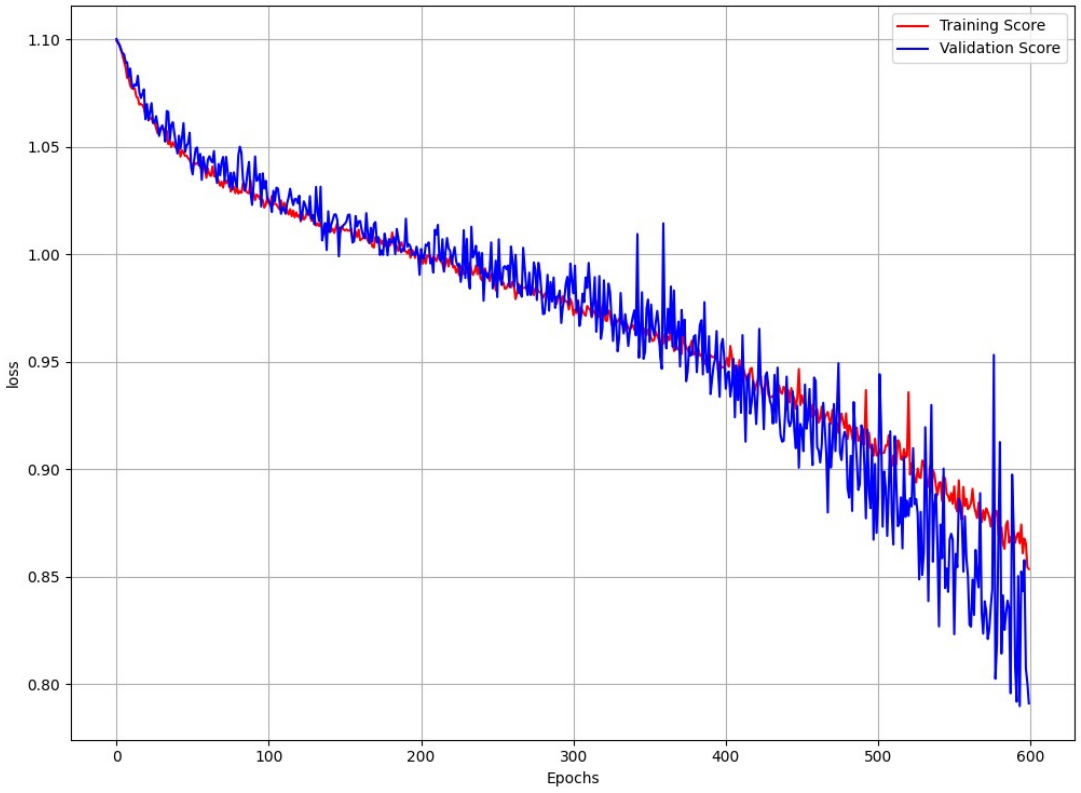
We can see clearly that model #8 had the best result as that of #10. There were no further tests with NLLLoss Loss function as the results were the same as with CrossEntropyLoss . The number of epochs that was used is 600.

When the model consists of only 2 hidden layers we can see that we have the best model with learning rate = 1e-1 which also gave the highest scores and in the other tests. The paradox that happens only with test#3 is that when we have increased out learning rate to 1e-3 we have bigger f1_score than precision.

When we have 3 hidden layers we can see that it performs better when they are bigger in size and have learning rate = 1e-1. This can be explained by the fact that we put weights in our classes for

the loss function and because of the low learning rate and the big size of the input we need more weights to be adjusted between the neurons.

Here we can see the loss vs epochs plot:



And the ROC curve plot: