NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

Student's full-name: Spanos Nikolaos
Academic Number: 7115112100023
Teacher: Koumparakis Manolis

2nd academic homework exercise of course Artificial Intelligence II
Academic year: 2022-2023
Master of Science in Computer Science

Athens, December 2022

Table of Contents

`

Sentiment analysis on imdb movie reviews using Deep Learning and PyTorch

The code is thoroughly developed in the Google Colab notebook (Project_II.ipynb) included in the project deliverables. This report will provide a thorough description of the results and the steps followed to develop and evaluate different model classifiers for the sentiment analysis on imdb movie reviews.

1. Importing the dataset

The dataset used is the exact same movies reviews from HW1. Thus, the transformation and preparation of the text sentences and the target sentiment for NLP experiments has already been developed in the previous Homework. On this homework the dataset is already cleaned and prepared for Natural Language Preprocessing[1].

2. Preparing data for PyTorch models

In this unit I describe the following units:

1) Create vocabulary from corpus and GloVe embeddings
2) Data Vectorization
3) Vector Padding
4) Data split into train, test, validation samples
5) Data split into batches

The experiments are separated into two categories:

1st category: The experiments using trainable word embeddings and custom build vocabulary.

2nd category: The experiments using GloVe word embeddings and vocabulary.

Each of the two categories has a three set of experiments, with each experiment testing a different neural network structure.

The experiments can be grouped like below:

1) Trivial Complexity Neural Network – Baseline
2) Advanced Complexity Neural Network

[1] Lemmatization, punctation removal, stop words removal, abbreviation transformation, shuffling, stratification.

Each group has a different network structure. The variations in the model structure will explained later in the report.

2.1 Create a vocabulary

The vocabulary built from a corpus or imported from pre-trained models is a very important component in the training of neural networks. Basically, the vocabulary is the first component to build in this pipeline. A vocabulary with the most important words can be very deterministic into training an accurate model classifier than a vocabulary with too many noises and words.

For building custom and trainable word embeddings I have used, from the 104,158 words in the dataset, only those unigrams and bigrams with minimum frequency of 100 times. Words that were found less than 100 times in the corpus were excluded from the vocabulary. Thus, creating a vocabulary of 7,043 words. For the first category of experiments with trainable embeddings I have used those 7,043 words. Whereas for the second category I used the 400,000 words imported from the pre-trained GloVe embeddings.

2.2 Data Vectorization

Next is the data vectorization. The transformation of words into a representation from an integer value. In the vectorization process I have taken into consideration the unknown words to the vocabulary by representing them with the index 0. This representation was applied on both categories of experiments. The sentences have been transformed from a sequence of tokens/words into a sequence of integers.

2.3 Vector Padding

Before importing the data samples into a Feed Forward neural network, it's important to fix the lengths of each vector to a constant value. The vector padding is an important process that could affect the model results. We don't want to lose information from selecting a very small value, like 25 words per sentence, or select a very high value to that would possibly let noise inside the sentence. For this step I have calculated the average sentence length by measuring the number of words in a sentence. Then, I calculated the 90% of the average length. This value was 235 words per sentence. I have reduced this value down to 150 words per sentence. In most of the experiments I have used 150 words per sentence while in a few of them I have used 235 words.

2.4 Data split into train, test, validation samples

Now that I have padded the sample vectors to fixed size, next I split the dataset into training, validation and test sets. The ratio used:

- Training set: 85%
- Validation set: 5% of training set
- Test set: 10%

It's important after the split to preserve the ratio of sentiment labels in the data samples. His is achieved after applying a stratified split.

2.5 Data split into batches

Finally, before starting the experimentation, the sentences and the target sentiment are grouped into batches. The batches are generated from a PyTorch method called DataLoader(). The latter will create shuffled batches in the form (sentence, sentiment). The number of sentences per batch is set to 64. Which translates to*, "Backpropagation will update the weights of the neural network every 64 sentences of training"*.

3. Feed forward neural networks - development, training, and evaluation

In this unit, I present the results from different experiments using either trainable embeddings or GloVe embeddings. The hyper-parameters of each experiment were monitored and selected from the Optuna[2] framework.

The hyper-parameters tuned are:

- Embedding's dimension in the case of trainable word-embeddings [*10 - 100*].
- Linear hidden units of the Linear dense layers [*8 - 128*].
- Dropout rate probability in the case of Dropout layer use [*0.25 – 0.75*].
- Activation function [*ReLU, Tanh*].
- Optimizer class [*Adam, RMSprop, SGD*].
- Learning rate [*0.001 - 0.01*].

Optuna works with trials. Each trial is a different combination of hyper-parameter values. The number of trials used is 16. Thus, 16 models have been trained on each experiment. After the training if 16 models,

[2] Optuna - A hyperparameter optimization framework

the best one of them is selected. The selection is made only by picking the model with the lowest validation loss. After a trial is finished the learning curves of training and validation losses are printed.
After the model selection, the one with the lowest validation loss is used to create an evaluation report with metrics and a confusion matrix. And finally, the ROC-AUC curve with the model predictions. The loss function used is the BCELoss() [Binary Cross Entropy] on the outputs of a Sigmoid() activation function to match the binary classification problem. All trials have been trained for 150 epochs with an Early Stopping mechanism of patience 5 and minimum delta/tolerance of 0.015.

3.1 Trivial Complexity models

This category of models includes simple neural networks with only one hidden layer. The model structure includes:

- Embedding layer
- Input linear layer (1 hidden layer)
- Output linear layer

Below is the model structure from one of the trials of Optuna framework for a trivial-complexity model with trainable embeddings.

SentimentClassifier_trivial_model_complexity(
(embedding): Embedding(7043, 78, padding_idx=0)
(input_layer): Linear(in_features=11700, out_features=84, bias=True)
(flatten_layer): Flatten(start_dim=1, end_dim=-1)
(output_layer): Linear(in_features=84, out_features=1, bias=True)
(activation): ReLU()
(activation_output): Sigmoid()
)

At each sub-unit (i.e., 3.1.1, 3.1.2, etc) I will report the best model hyperparameters selected from Optuna, the learning curves, the confusion matrix, and the ROC-AUC curve of the model. At the end of the unit, I will present a table with the results of the best models from each experiment to make a comparison and select the best one.

3.1.1 Trainable word-embeddings

==========
Best model parameters:

==========

activation: Tanh

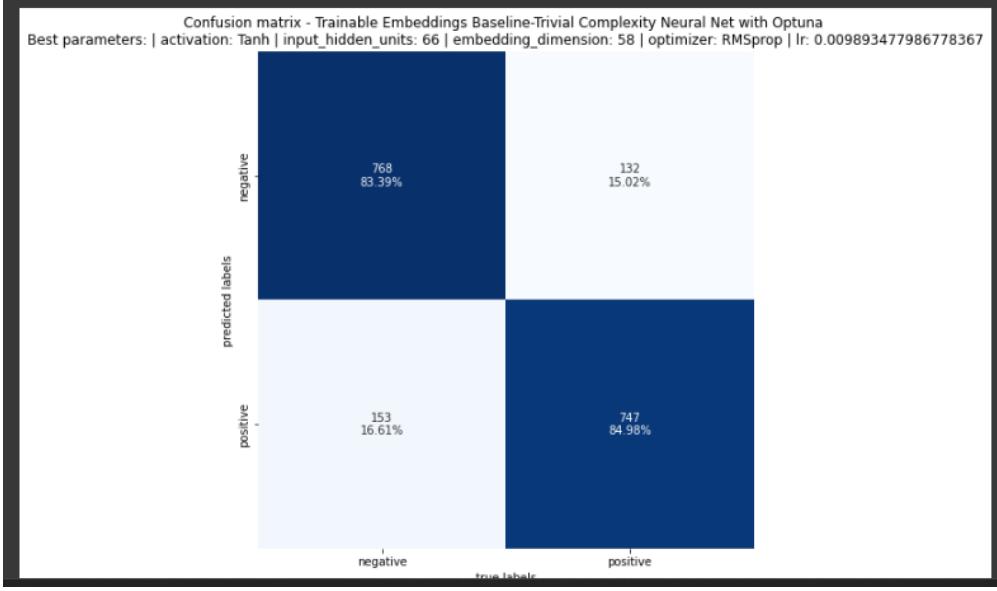
input_hidden_units: 66

embedding_dimension: 58

optimizer: RMSprop

lr: 0.009893477986778367

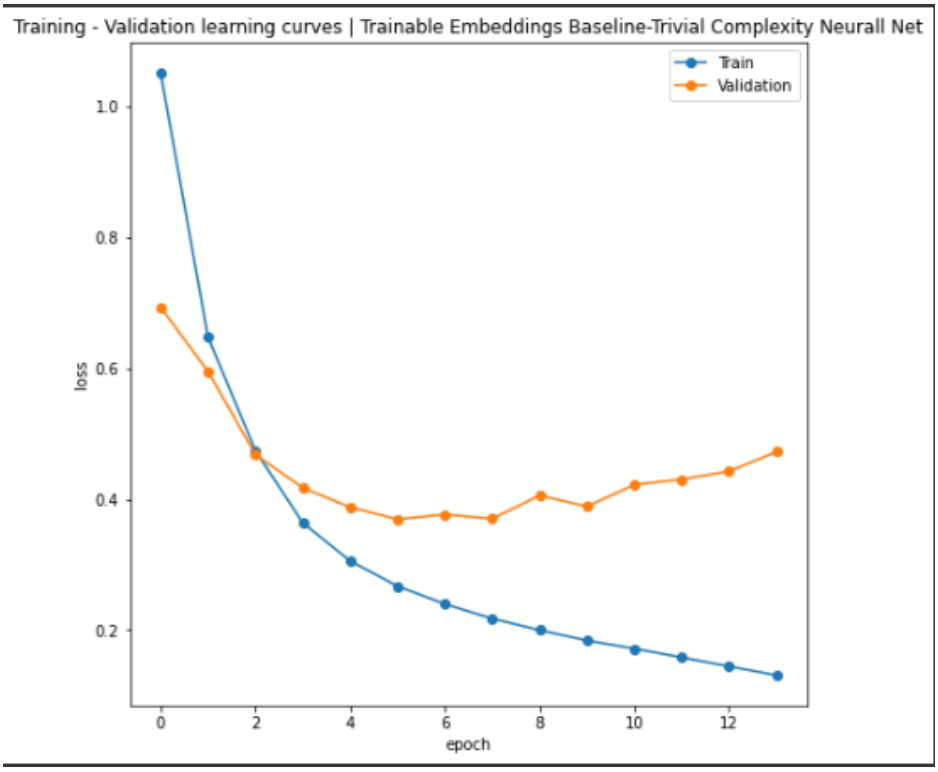
```
Evaluation metrics: Trainable Embeddings Baseline-Trivial Complexity Neural Net with Optuna
Best parameters: | activation: Tanh | input_hidden_units: 66 | embedding_dimension: 58 | optimizer: RMSprop | lr: 0.009893477986778367

precision score: 0.8498
recall_score: 0.8300
roc score: 0.8417
f1_score: 0.8398

model bias: 0.0002
model variance: 0.2498999983072281

              precision    recall  f1-score   support

    negative       0.83      0.85      0.84       900
    positive       0.85      0.83      0.84       900

    accuracy                           0.84      1800
   macro avg       0.84      0.84      0.84      1800
weighted avg       0.84      0.84      0.84      1800
```



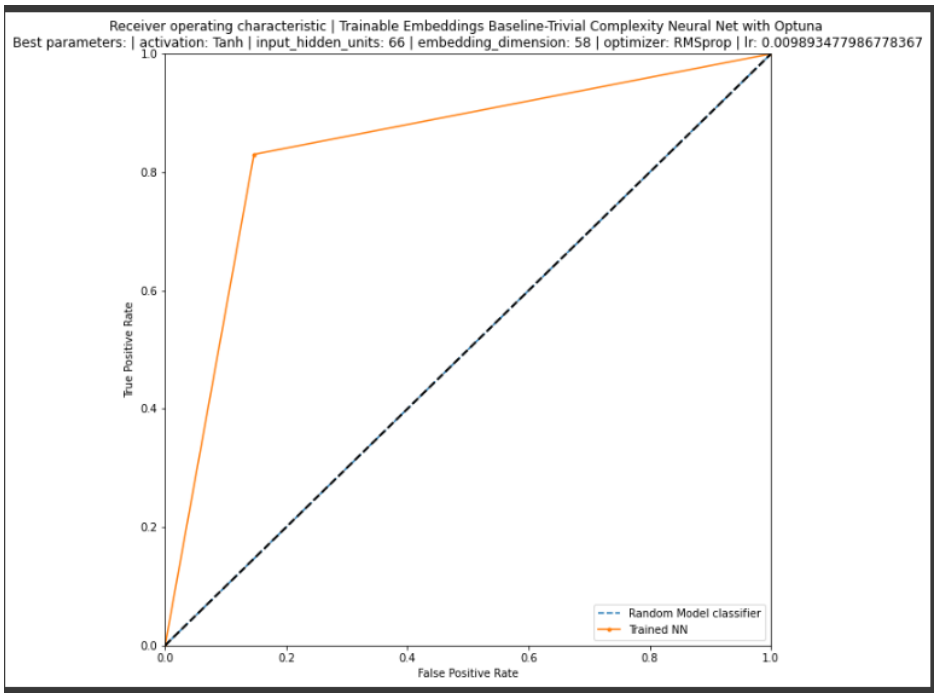
1-Evaluation Report | Trainable Embeddings-Trivial Complexity

FPR (False Positive Ratio): 16.61%

FNR (False Negative Ratio): 15.02%



2-Learning Curve | Trainable Embeddings - Trivial Complexity



3-ROC Curve | Trainable Embeddings - Trivial Complexity

3.1.2 GloVe word-embeddings

GloVe:

- "6B": http://nlp.stanford.edu/data/glove.6B.zip
- 50 embedding dimensions

SentimentClassifier_trivial_model_complexity_glove(

(embedding): Embedding(400000, 50, padding_idx=0)

(input_layer): Linear(in_features=50, out_features=84, bias=True)

(flatten_layer): Flatten(start_dim=1, end_dim=-1)

(output_layer): Linear(in_features=84, out_features=1, bias=True)

(activation): ReLU()

(activation_output): Sigmoid()

)

Please note that the weights of the pre-trained GloVe embeddings are freezed and not trained.

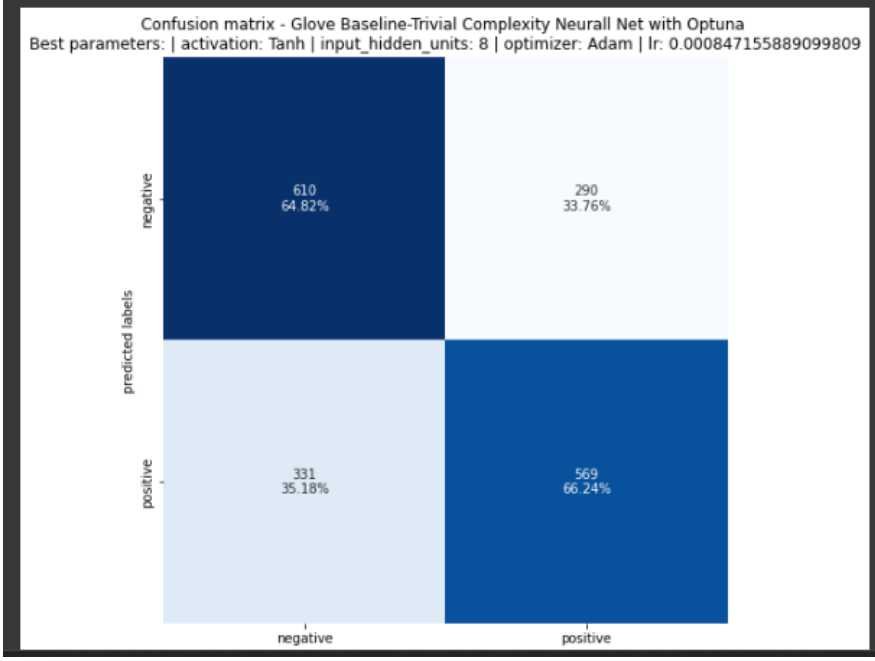
==========
Best model parameters:
==========
activation: Tanh

input_hidden_units: 8

optimizer: Adam

lr: 0.000847155889099809

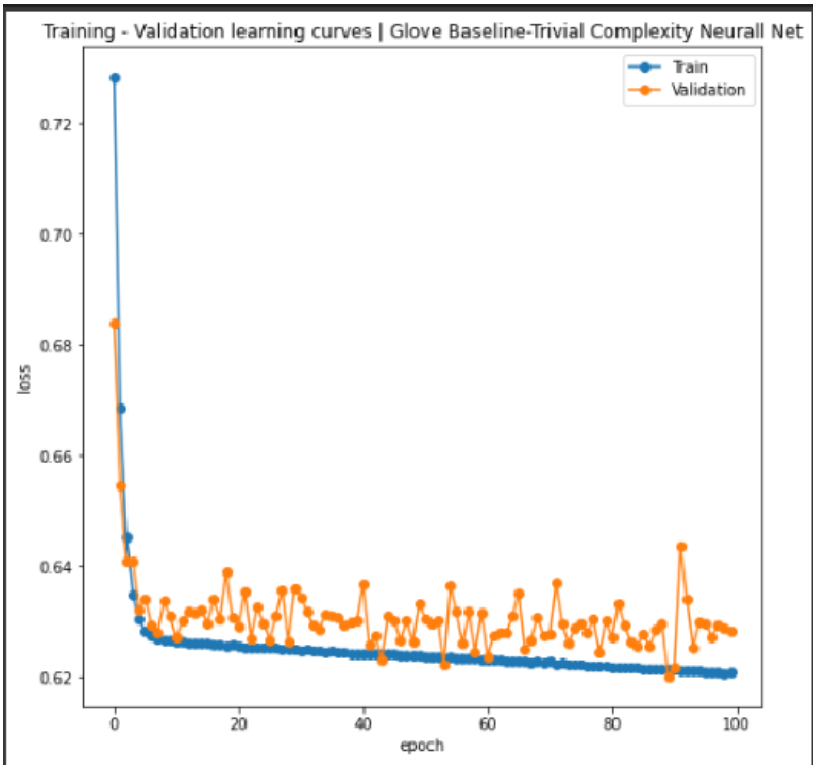
```
Evaluation metrics: Glove Baseline-Trivial Complexity Neurall Net with Optuna
Best parameters: | activation: Tanh | input_hidden_units: 8 | optimizer: Adam | lr: 0.000847155889099809

precision score: 0.6624
recall_score: 0.6322
roc score: 0.6550
f1_score: 0.6470

model bias: 0.001
model variance: 0.24950000643730164

              precision    recall  f1-score   support

    negative       0.65      0.68      0.66       900
    positive       0.66      0.63      0.65       900

    accuracy                           0.66      1800
   macro avg       0.66      0.66      0.65      1800
weighted avg       0.66      0.66      0.65      1800
```



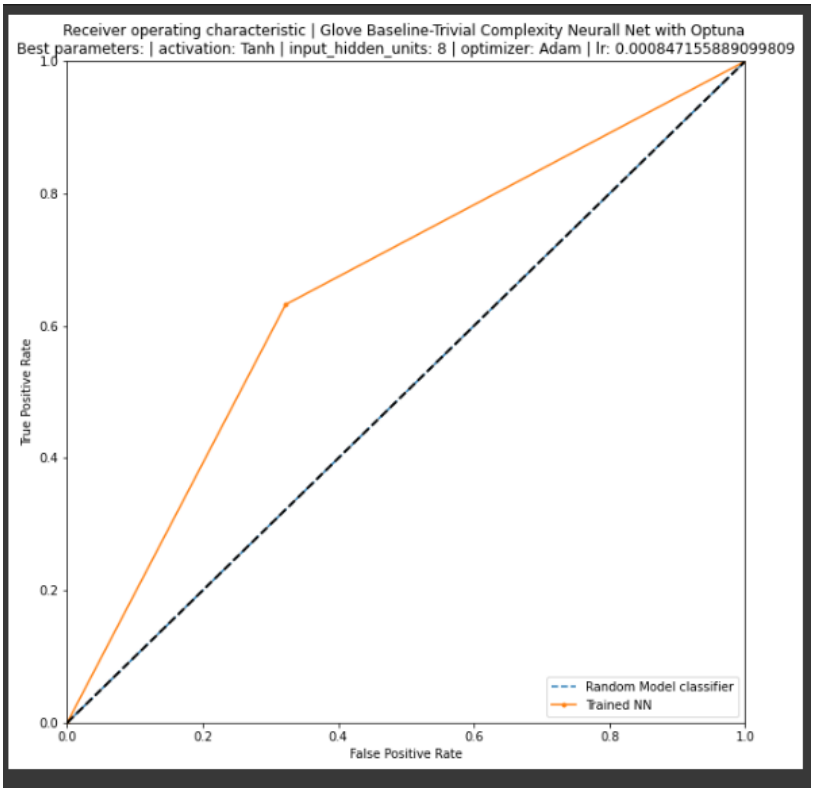
4-Evaluation Report | GloVe Embeddings-Trivial Complexity

FPR (False Positive Ratio): 35.18%

FNR (False Negative Ratio): 33.76%



5-Learning Curve | GloVe Embeddings - Trivial Complexity



6-ROC Curve | GloVe Embeddings - Trivial Complexity

3.1.3 GloVe word-embeddings (three mathematical transformations on embeddings)

In the third and final experiment, on the trivial complexity models, I have applied three mathematical transformations on the GloVe embeddings layer of the model. Specifically,

- Average of the embeddings.
- Maximum value of the embeddings.
- Minimum value of the embeddings.

The key to understand how this process works is to keep in mind that every sentence in the batch is a representation of a vector with fix length 150. Each token in the vector represents a word of the sentence. Each token/word is represented by a vector of 50 values (1 embedding). Thus, the size of a sentence in the batch is [1,150,50]. By applying each of the above three mathematical transformations we get back a batch of size [1,50]. The 150 words are basically downsized to 1 dimension to represent the whole sentence. So, after applying the average(mean) function we return a single sentence with a vector of 50 values. With each value representing the average of the 150x50 words. The same philosophy applies for maximum and minimum values.

With those three transformations a concatenated input layer is constructed to hold information from the three mathematical expressions. Each a way to feature engineer more features for the model.

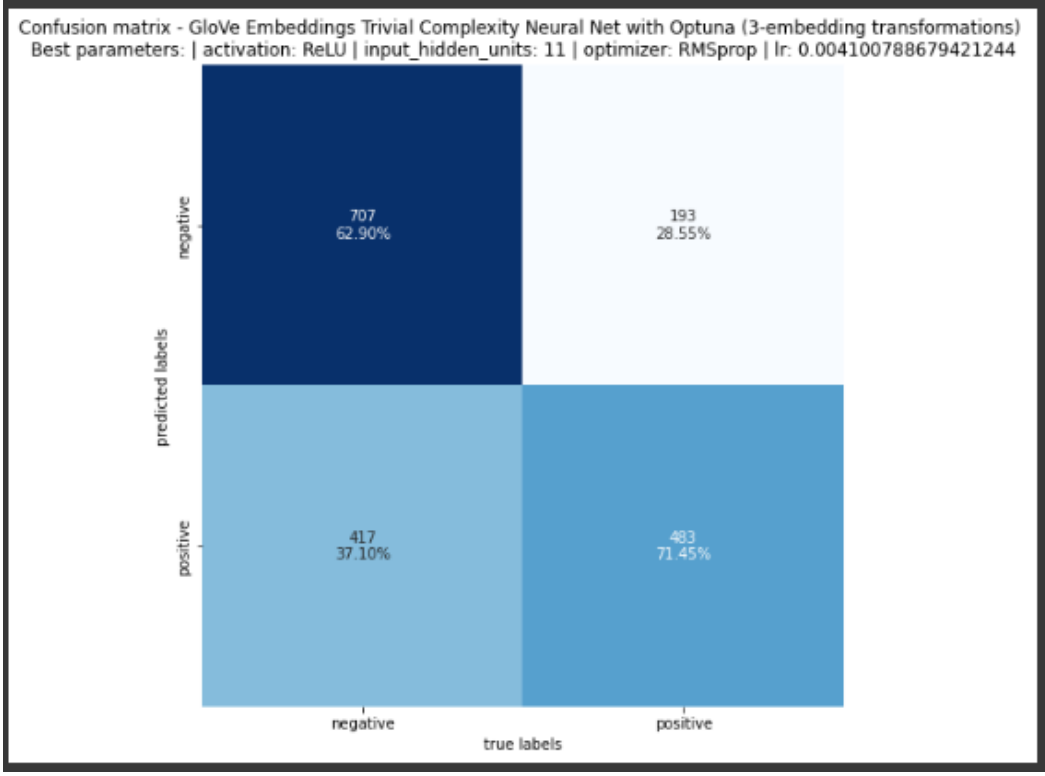
==========
Best model parameters:
==========
activation: ReLU

input_hidden_units: 11

optimizer: RMSprop

lr: 0.004100788679421244

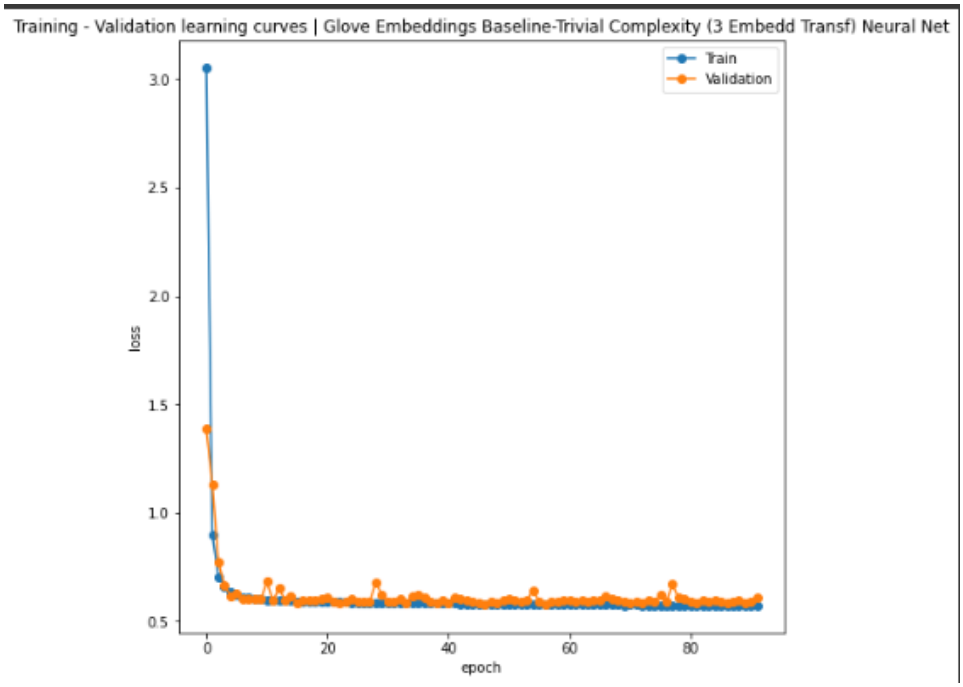
```
Evaluation metrics: GloVe Embeddings Trivial Complexity Neural Net with Optuna (3-embedding transformations)
Best parameters: | activation: ReLU | input_hidden_units: 11 | optimizer: RMSprop | lr: 0.004100788679421244

precision score: 0.7145
recall_score: 0.5367
roc score: 0.6611
f1_score: 0.6129

model bias: 0.031
model variance: 0.2345


              precision    recall  f1-score   support

    negative       0.63      0.79      0.70       900
    positive       0.71      0.54      0.61       900

    accuracy                           0.66      1800
   macro avg       0.67      0.66      0.66      1800
weighted avg       0.67      0.66      0.66      1800
```



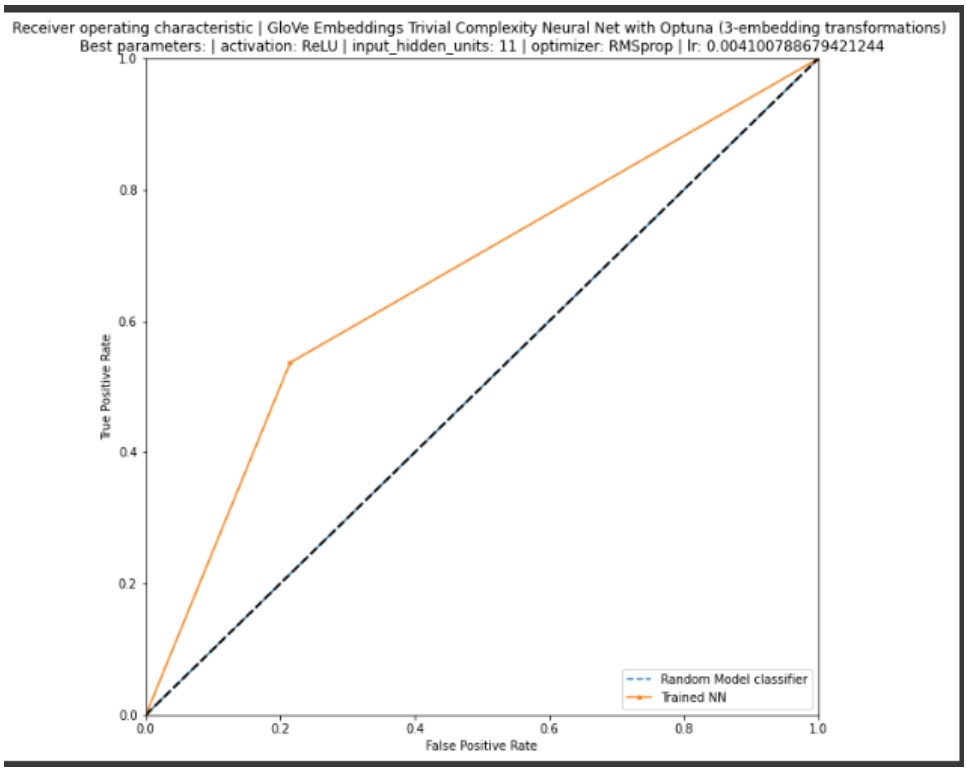
7-Evaluation Report | GloVe Embeddings (three expressions)-Trivial Complexity

FPR (False Positive Ratio): 37.10%

FNR (False Negative Ratio): 20.55%



8-Learning Curve | GloVe Embeddings (three expressions) - Trivial Complexity



9-ROC Curve | GloVe Embeddings (three expressions) - Trivial Complexity

3.1.4 Evaluation metrics table

Below I report the evaluation metric results from the three trivial complexity experiments and I also comment of the results from all the previous figures to make a valid conclusion on the best trivial complexity model.

Model name	Precision score	Recall score	Roc score	F1-score	Bias	Variance
Trainable embeddings	0.8498	0.8300	0.8417	0.8390	0.002	0.2498
GloVe embeddings	0.6624	0.6322	0.6550	0.6470	0.001	0.2495
GloVe embeddings (2)	0.7145	0.5367	0.6611	0.6129	0.031	0.2345

1-Evaluation metric results - Trivial complexity models

With green is marked the metric with the best performance over the three models. With orange is marked the GloVe with the best performance from the other GloVe models.

Overall, based on the selected scores, the neural network trained on a 7,043 words vocabulary had the best prediction performance over both models trained with the GloVe embeddings as input. Also, the False Positive and False Negative ratios for the trainable embeddings model is much lower than the other two. Another conclusion that we can derive, which is true for all the three models, is that they stopped learning around the epochs 10-20. This can be observed from the reported learning curves. For the first model the training is stopped from the first 12 epochs, showing that the model started to overfit and stopped. Next, the bias values are low with the first and second models achieving a nearly 0.0 bias value. Showing that underfitting is out-of-question. However, the learning curve of the second model shows that the predictions are mostly random, and the classifier couldn't capture the data pattern. Finally, the variance values are also low in all the three models, with the third model achieving the lowest value. The reported variance values could indicate a slightly overfitted classifiers. Especially, in the case of the first model.

3.2 Advanced Complexity models

This category of models includes complex neural networks with more than one hidden layer, normalization layer, dropout layers and regularization. The model structure includes:

- Embedding layer
- Input linear layer (1-5 hidden layers)
- BatchNormilaztion layer
- Dropout layer

- Output linear layer
- Additional hyper-parameter (number of Linear hidden layers)

Below is the model structure from one of the trials of Optuna framework for an advanced-complexity model with trainable embeddings.

SentimentClassifier_advanced_complexity_optuna_v2(

(embedding): Embedding(7043, 11, padding_idx=0)

(input_layer): Linear(in_features=11, out_features=127, bias=True)

(flatten_layer): Flatten(start_dim=1, end_dim=-1)

(normalize_input_layer): BatchNorm1d(127, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(dropout_input_layer): Dropout(p=0.6734758924923008, inplace=False)

(output_layer): Linear(in_features=8, out_features=1, bias=True)

(activation): ReLU()

(activation_embeddings): Tanh()

(activation_output): Sigmoid()

(fully_connected_layer_0): Linear(in_features=127, out_features=8, bias=True)

(fully_connected_layer_1): BatchNorm1d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(fully_connected_layer_2): ReLU()

(fully_connected_layer_3): Dropout(p=0.6734758924923008, inplace=False)

)

3.2.1 Trainable word-embeddings

==========
Best model parameters:
==========
activation: ReLU

input_hidden_units: 127

embedding_dimension: 11

dropout_rate: 0.6734758924923008

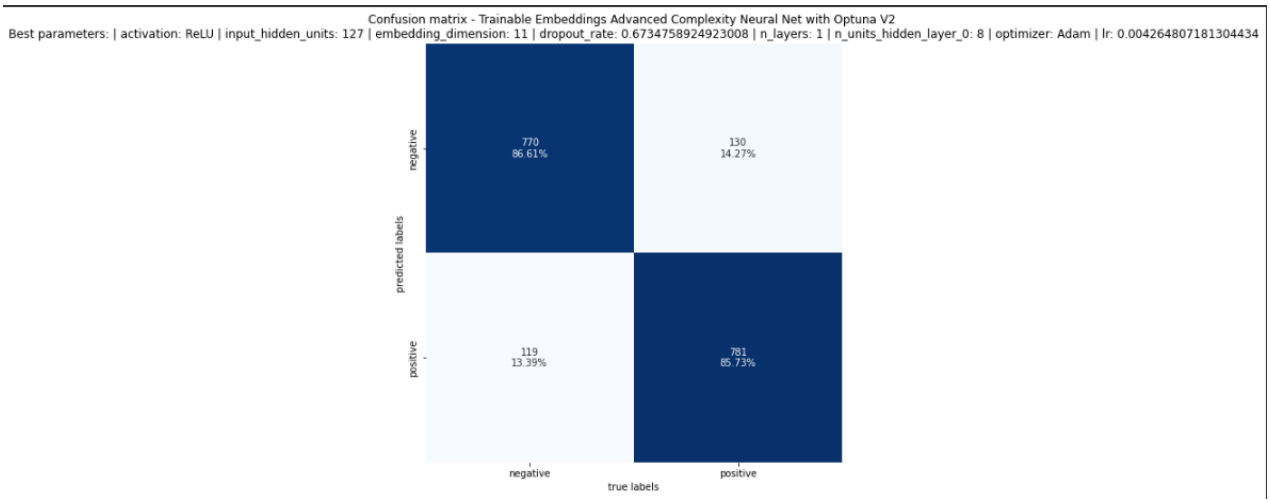
n_layers: 1

n_units_hidden_layer_0: 8

optimizer: Adam

lr: 0.004264807181304434

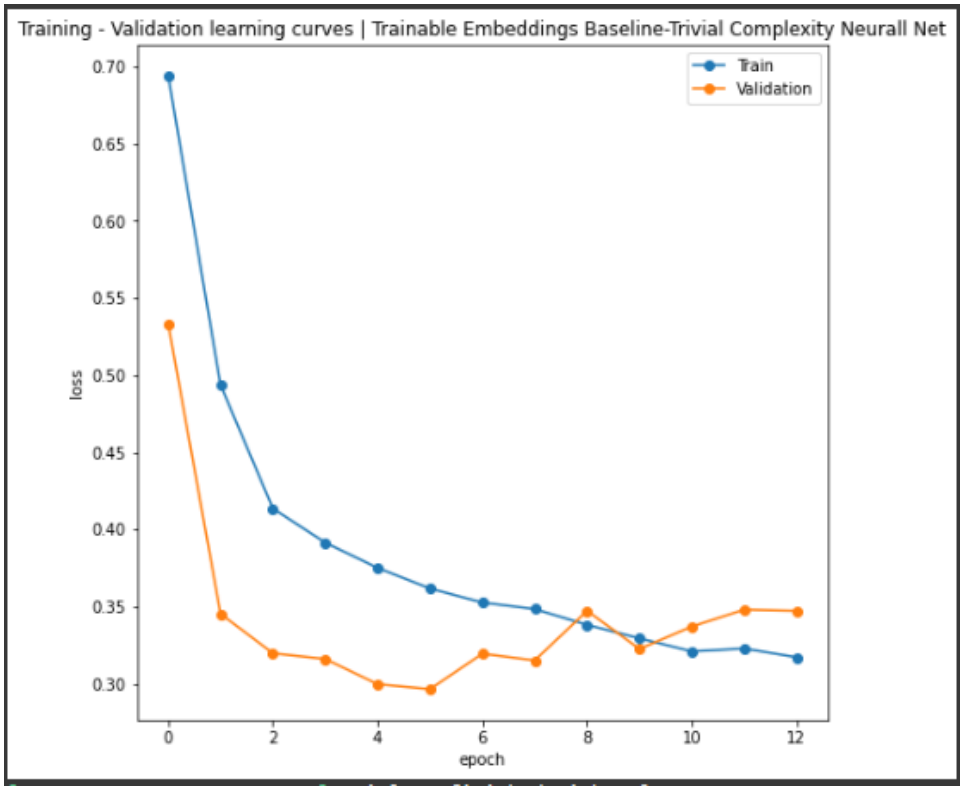
An obvious difference with the trivial-complexity model is the extra Linear hidden layer in the best model selected from Optuna with advanced complexity structure.



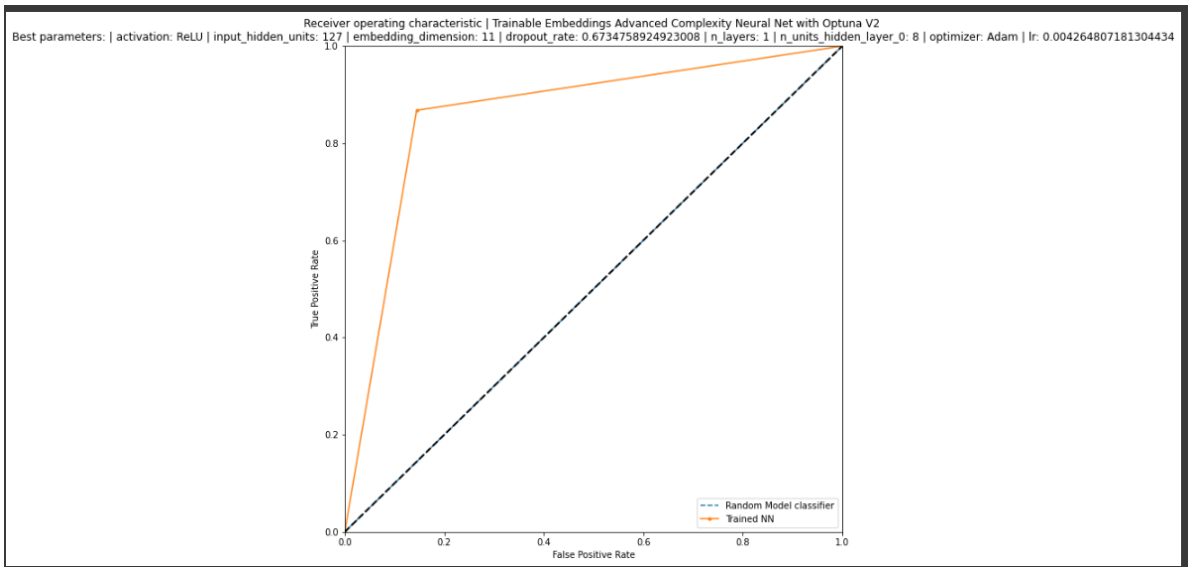
10-Evaluation Report | Trainable Embeddings-Advanced Complexity

FPR (False Positive Ratio): 13.39%

FNR (False Negative Ratio): 14.27%



11-Learning Curve | Trainable Embeddings - Advanced Complexity



12-ROC Curve | Trainable Embeddings-Advanced Complexity

3.2.2 GloVe embeddings (three mathematical transformations on embeddings)

==========
Best model parameters:
==========

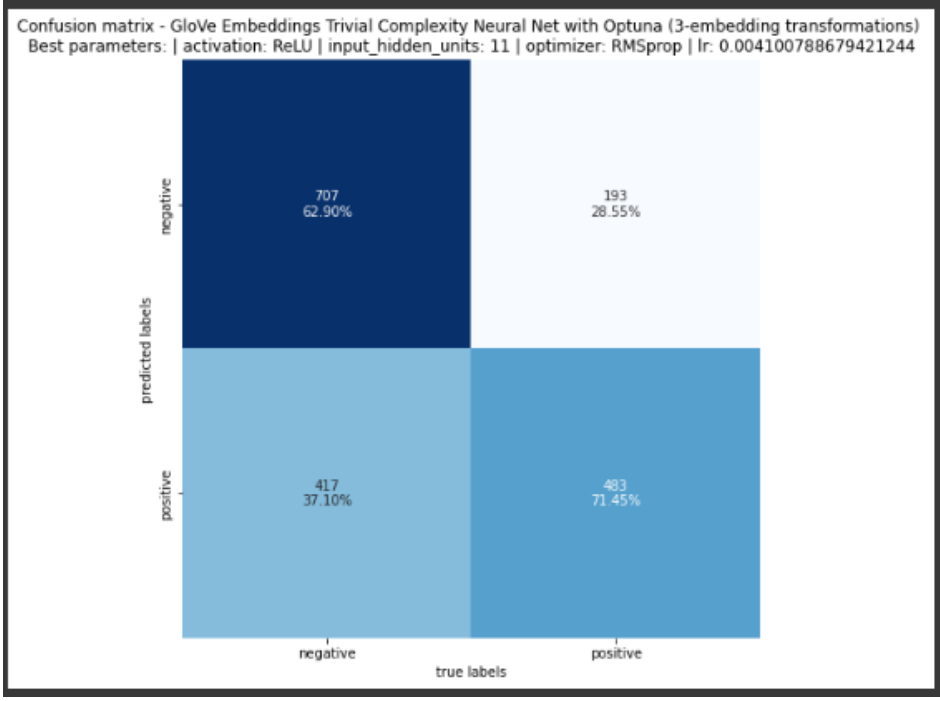
activation: ReLU

input_hidden_units: 11

optimizer: RMSprop

lr: 0.004100788679421244

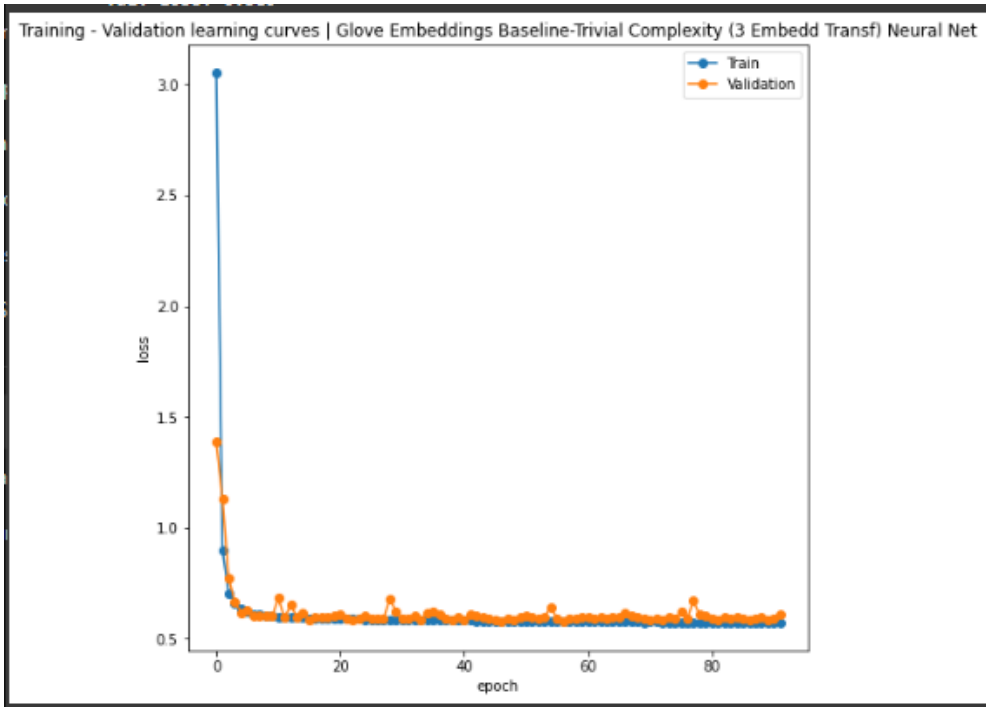
```
Evaluation metrics: GloVe Embeddings Trivial Complexity Neural Net with Optuna (3-embedding transformations)
Best parameters: | activation: ReLU | input_hidden_units: 11 | optimizer: RMSprop | lr: 0.004100788679421244

precision score: 0.7145
recall_score: 0.5367
roc score: 0.6611
f1_score: 0.6129

model bias: 0.031
model variance: 0.2345

              precision    recall  f1-score   support

    negative       0.63      0.79      0.70       900
    positive       0.71      0.54      0.61       900

    accuracy                           0.66      1800
   macro avg       0.67      0.66      0.66      1800
weighted avg       0.67      0.66      0.66      1800
```



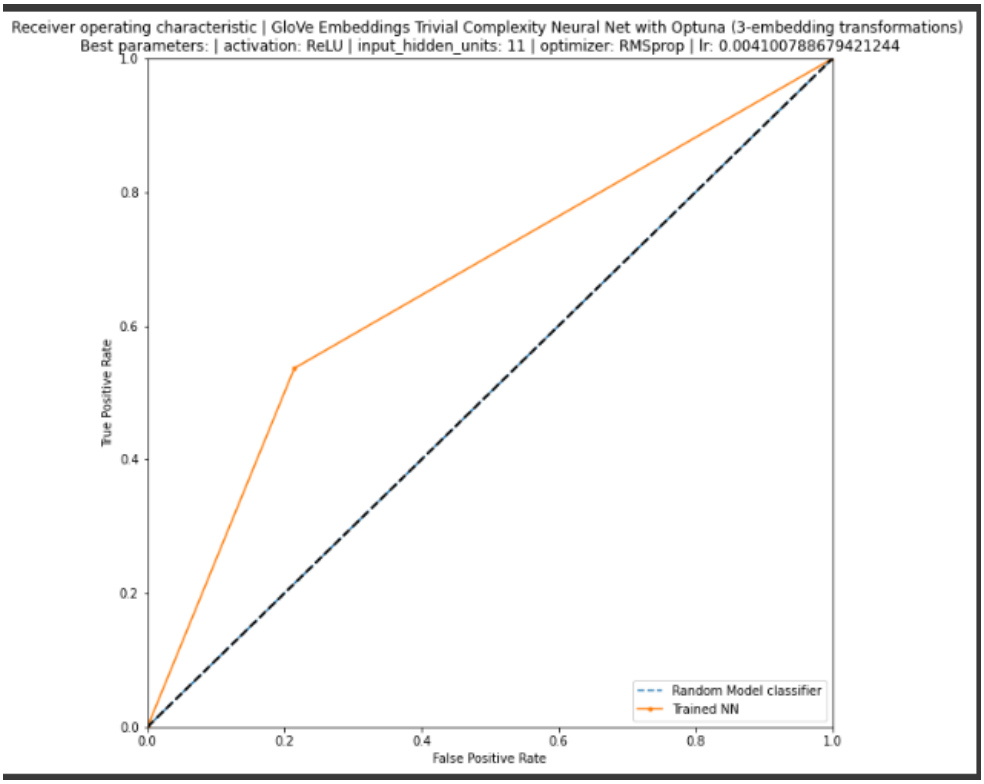
13-Evaluation Report | GloVe Embeddings-Advanced Complexity

FPR (False Positive Ratio): 37.10%

FNR (False Negative Ratio): 20.55%



14-Learning Curve | GloVe Embeddings - Advanced Complexity



15-ROC Curve | GloVe Embeddings-Advanced Complexity

3.2.3 GloVe embeddings (three mathematical transformations on embeddings) v2

This is the second version of the GloVe embeddings model of section 3.2.2

The slightly different neural network has the following differences from its predecessor:

- Slight increase in hidden layer units per linear layer. From [64,128] to [128, 256] and from [8,64] to [8,128].
- Increased total words per sentence from 150 to 235 (represents 90% of the length of all sentences in the sample).
- Added weight decay (l2 penalty) on the learning rate of the Optimizers.

==========
Best model parameters:
==========
activation: Tanh

input_hidden_units: 161

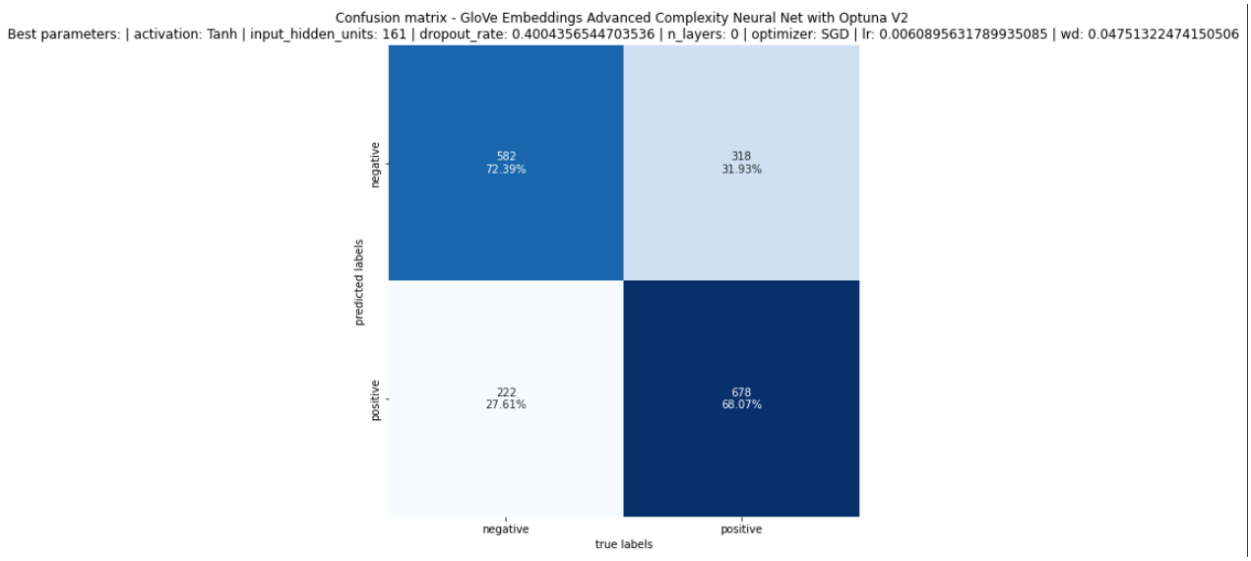
dropout_rate: 0.4004356544703536

n_layers: 0

optimizer: SGD

lr: 0.0060895631789935085

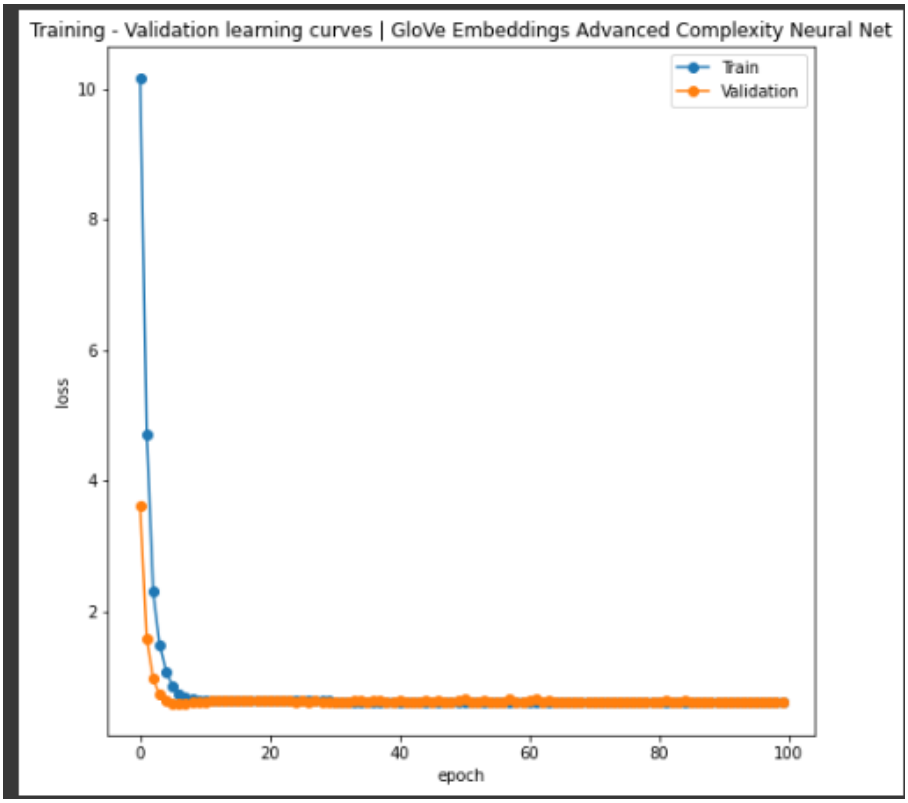
wd: 0.04751322474150506



16-Evaluation Report | GloVe Embeddings V2-Advanced Complexity

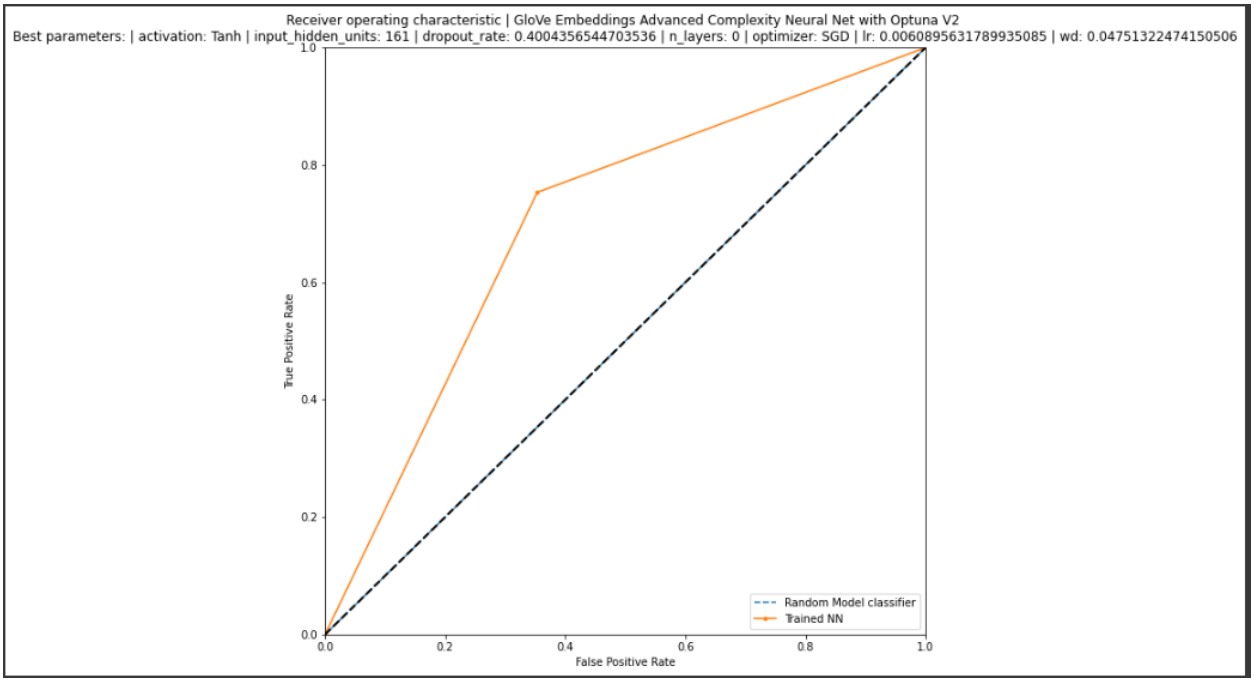
FPR (False Positive Ratio): 27.61%

FNR (False Negative Ratio): 31.93%



17-Learning Curve | GloVe Embeddings V2 - Advanced Complexity

Increasing the training parameters of the hidden layers and the total size of each sentence from 150 to 235 words, it's clear from the learning curve that the model classifier quickly learned the data pattern at around epoch 10 and after that epoch the model didn't not continue to learn.



18-ROC Curve | GloVe Embeddings V2-Advanced Complexity

3.2.4 GloVe embeddings (three mathematical transformations on embeddings) v3

This is the third version of the GloVe embeddings model of section 3.2.2.

The different neural network has the following differences from its predecessor:

- After each mathematical transformation of the embedding layers, I passed each embedding layer through the activation function Tanh().

==========
Best model parameters:
==========
activation: Tanh

input_hidden_units: 195

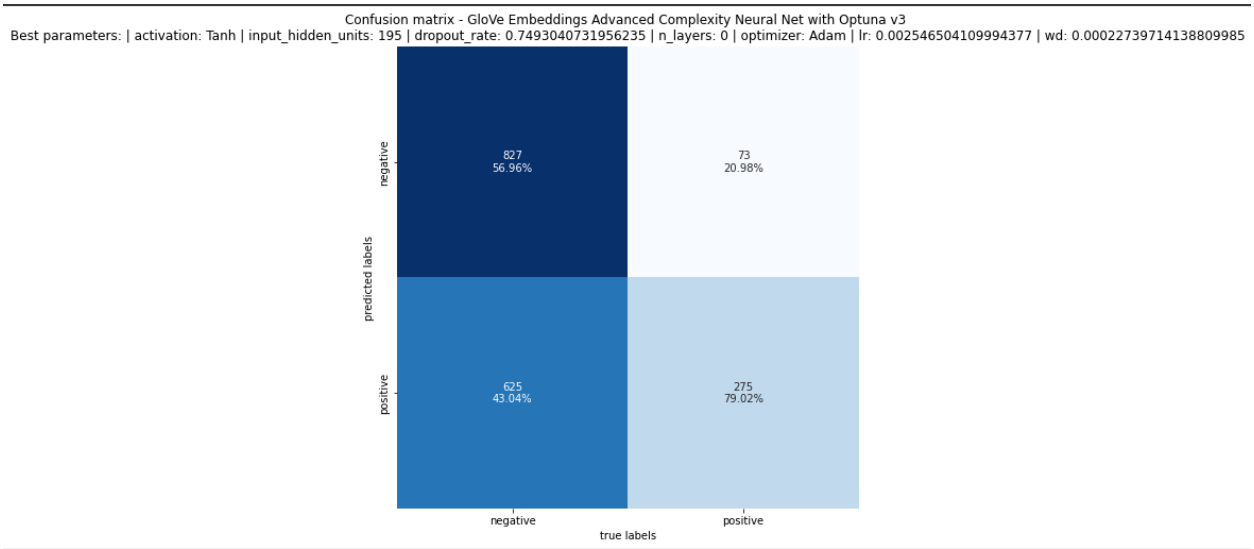
dropout_rate: 0.7493040731956235

n_layers: 0

optimizer: Adam

lr: 0.002546504109994377

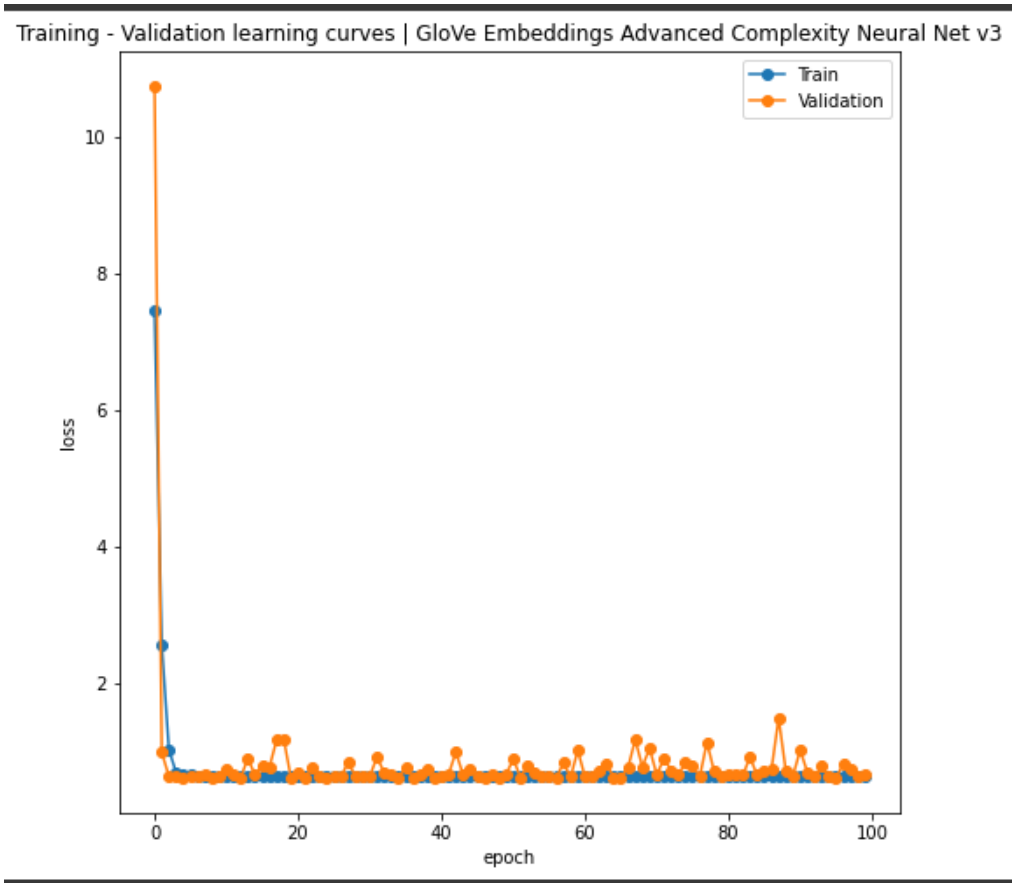
wd: 0.00022739714138809985



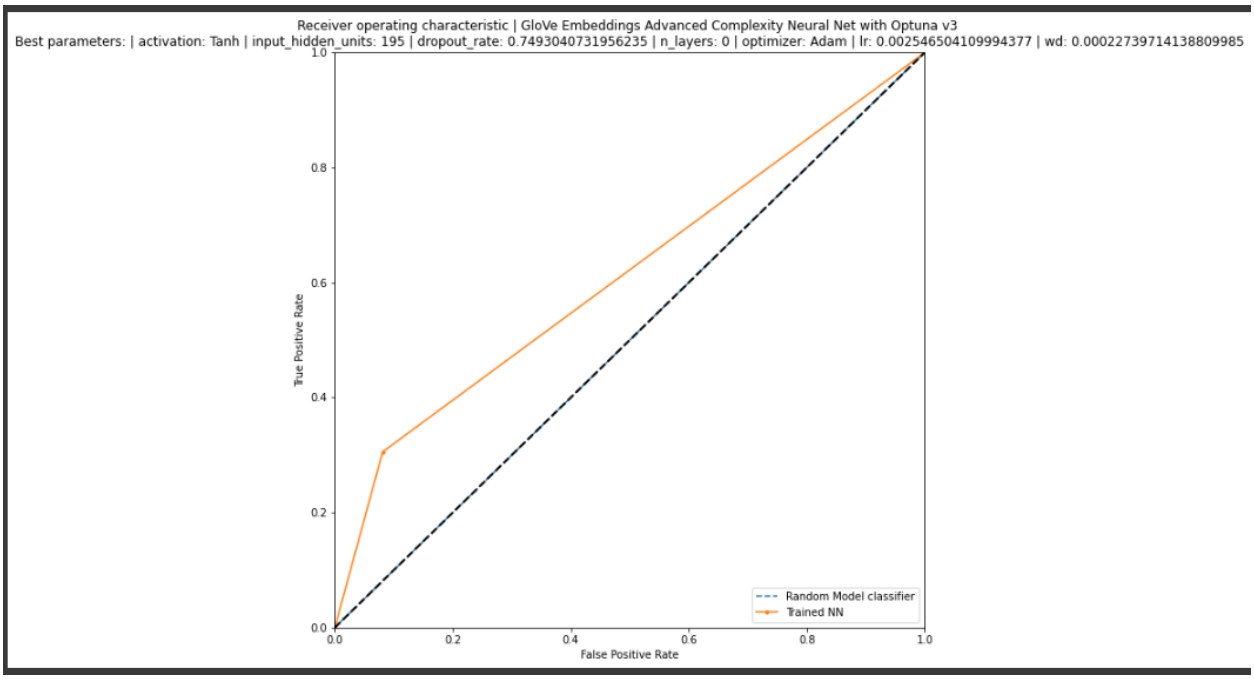
19-Evaluation Report | GloVe Embeddings V3-Advanced Complexity

FPR (False Positive Ratio): 43.04%

FNR (False Negative Ratio): 20.98%



20-Learning Curve | GloVe Embeddings V3 - Advanced Complexity



21-ROC Curve | GloVe Embeddings V3-Advanced Complexity

Totally off ROC curve. Very close to the predictions made by a random model and too many False Positives, reviews that were negative but misclassified as positive.

3.2.5 GloVe embeddings (average/mean transformation) v4

For the next experiment I have applied the following:

- I used the 100-dimensional GloVe embeddings matrix.
- 150 words per sentence
- 128 training batch size in the DataLoader() object.

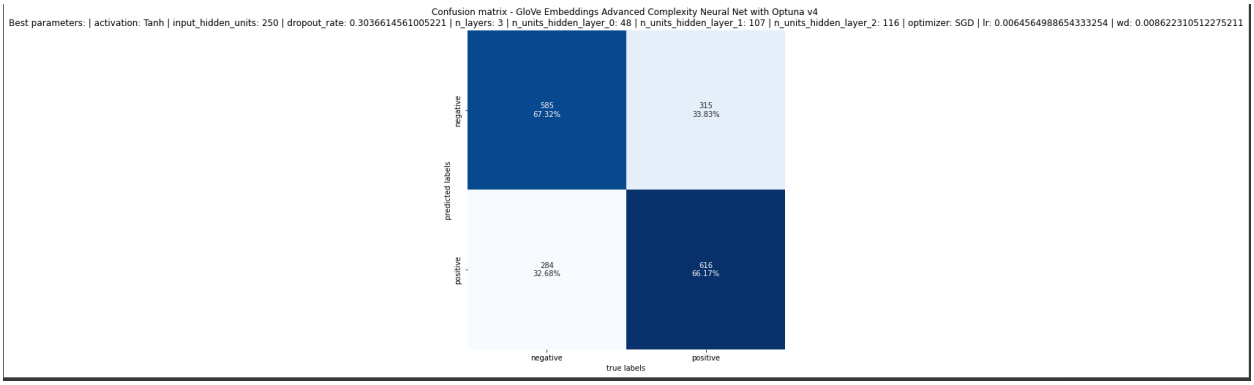


Figure 22-Evaluation Report | GloVe Embeddings V4-Advanced Complexity

FPR (False Positive Ratio): 32.68%

FNR (False Negative Ratio): 33.83%

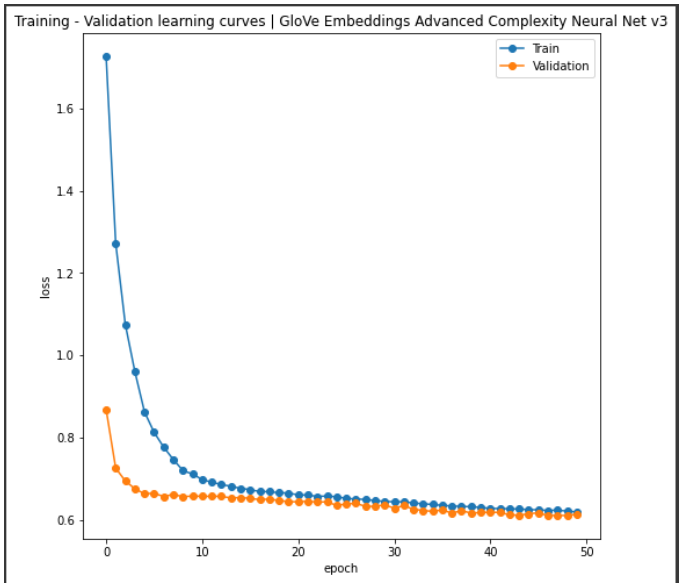


Figure 23-Learning Curve | GloVe Embeddings V4- Advanced Complexity

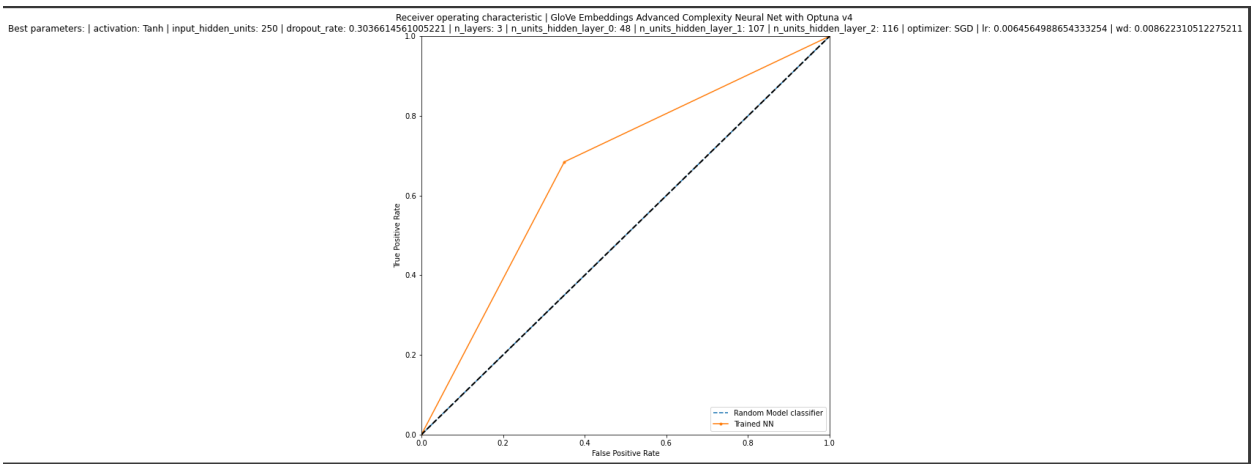


Figure 24-ROC Curve | GloVe Embeddings V4-Advanced Complexity

The increase in the embedding matrix and the decrease in the total number of words per sentence didn't have much of improvement in the predictive ability of the model.

3.2.6 GloVe embeddings (three transformations) v5

For the next experiment I have applied the following:

- Kept the same word-embeddings dimension (100)

- Increased words per sentence from 150 to 165 words (represents the 80% of sentence length in the dataset. 80% of the sentence have a sentence length of at least 165 words).
- Increased the training batch size from 128 to 256.
- Added the three mathematical transformations ([mean, max, min]) on word-embeddings.
- Increased total number of trials from 16 to 20.
- Increase number of epochs from 50 to 100.

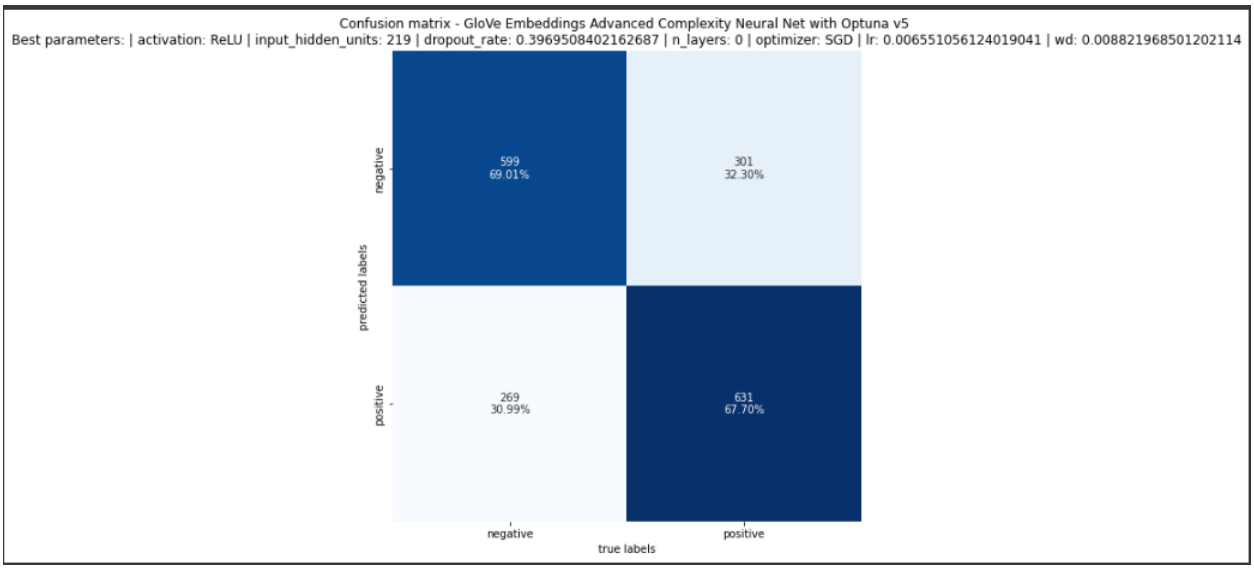


Figure 25-Evaluation Report | GloVe Embeddings V5-Advanced Complexity

FPR (False Positive Ratio): 30.99%

FNR (False Negative Ratio): 32.30%

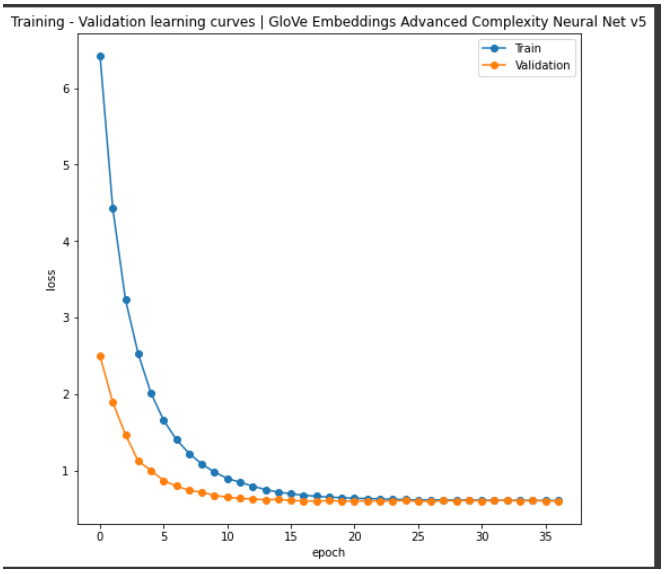


Figure 26-Learning Curve | GloVe Embeddings V5- Advanced Complexity

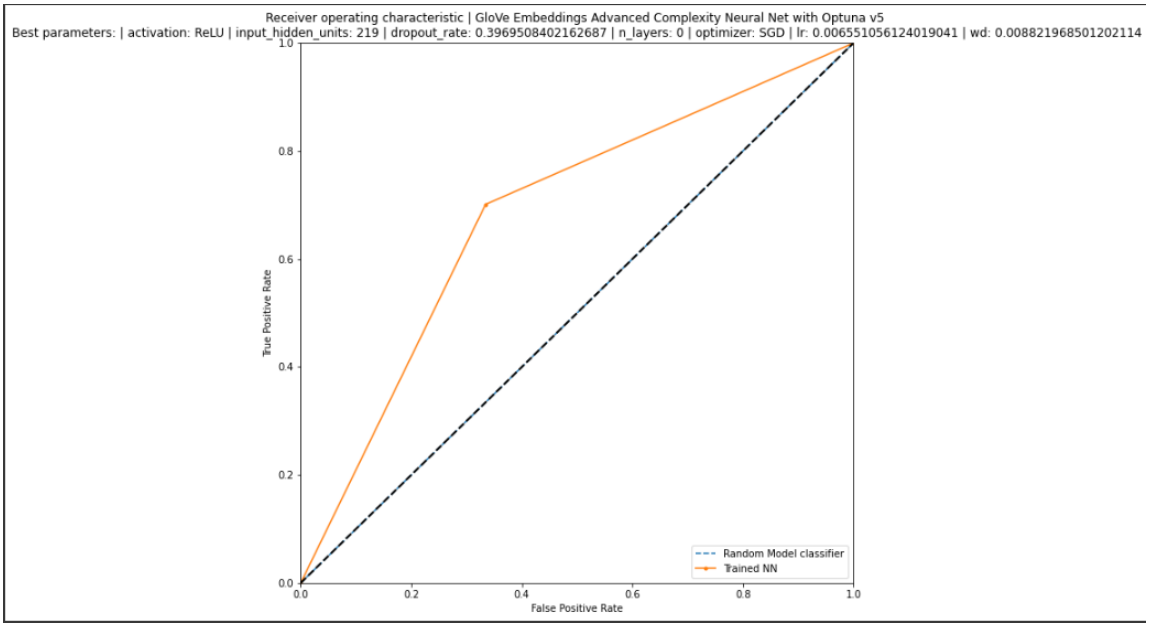


Figure 27-ROC Curve | GloVe Embeddings V5-Advanced Complexity

3.2.7 GloVe embeddings (three transformations) v6

For the next experiment I have applied the following changes. Always keeping in mind, the results of the previous experiments and adjusting the hyper-parameters.

- Increased GloVe embeddings dimension matrix from 100 to 200 vectors.

The rest of the changes were the same as in experiment 3.2.6 (check above).

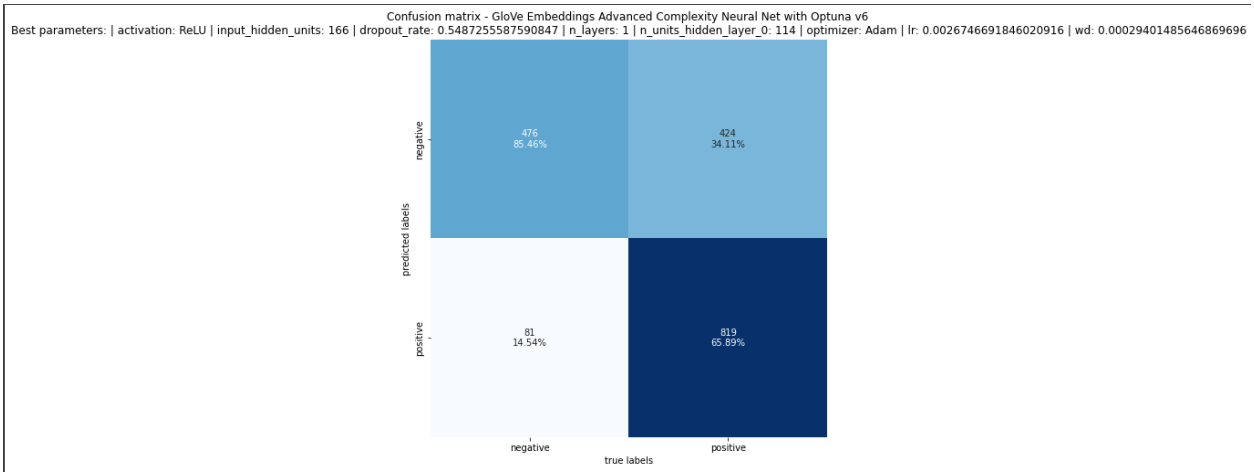


Figure 28-Evaluation Report | GloVe Embeddings V6-Advanced Complexity

FPR (False Positive Ratio): 14.54%

FNR (False Negative Ratio): 34.11%

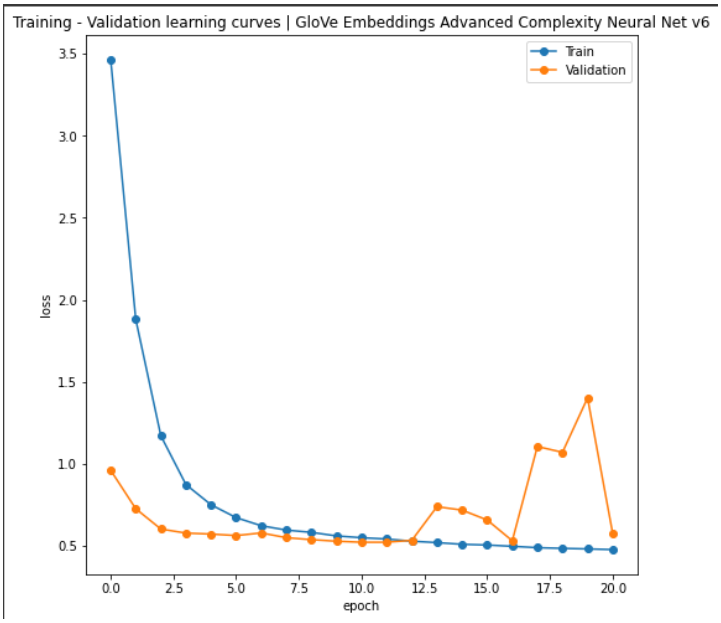


Figure 29-Learning Curve | GloVe Embeddings V6- Advanced Complexity

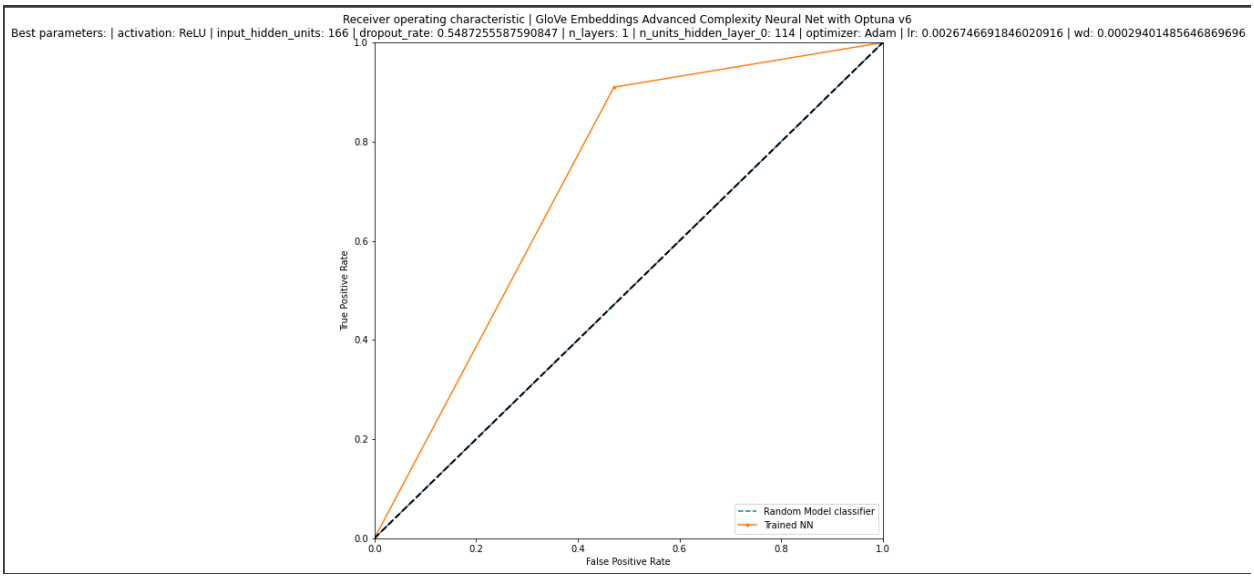


Figure 30-ROC Curve | GloVe Embeddings V6-Advanced Complexity

3.2.8 GloVe embeddings (three transformations) v7

For the next experiment I have applied the following:

- 200 embeddings dimension for GloVe embeddings.
- Increased trials from 16 to 25. Giving the ability to Optuna to create more hyper-parameters combinations.

- Increased validation size from 5% to 15%. This change was applied after spotting many fluctuations in the validation loss learning curve. Many ups and downs in the learning curve means that the model classifier didn't have enough data to make accurate predictions.
- Decreased the learning rate [0.0005, 0.001] to [0.00001, 0.0005].
- Decreased the weight_decay of optimizers from [0, 0.1] to [0, 0.0001].

The last two decreases were also an effort to flatten the validation learning curve.

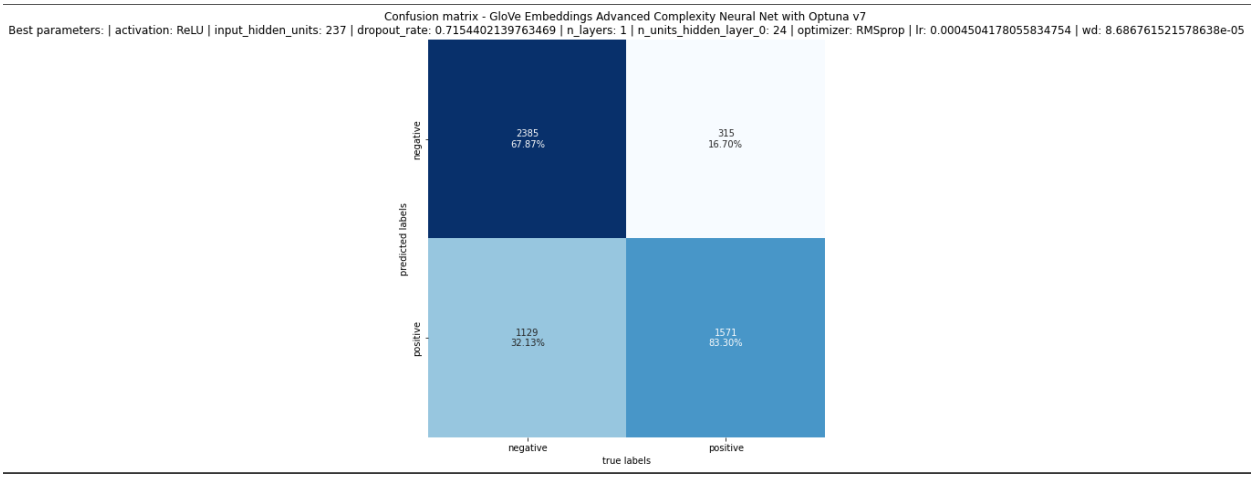


Figure 31-Evaluation Report | GloVe Embeddings V7-Advanced Complexity

FPR (False Positive Ratio): 32.13%

FNR (False Negative Ratio): 16.70%

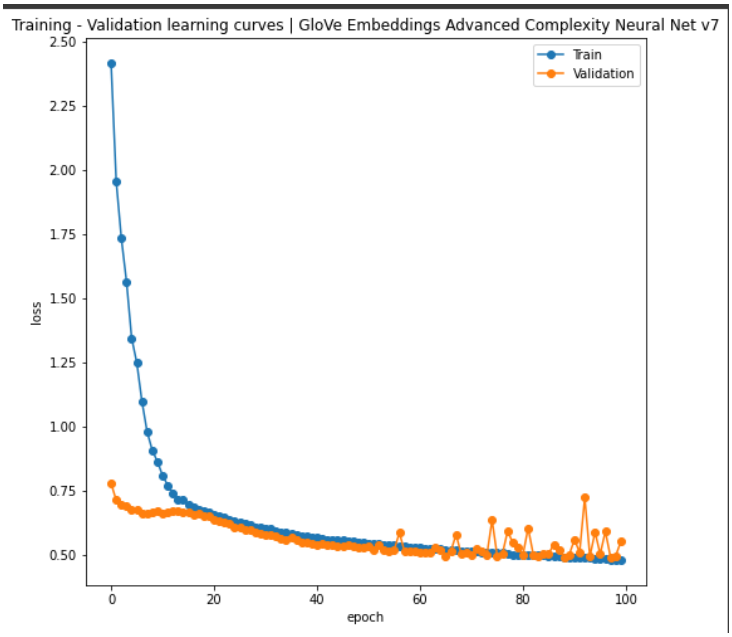


Figure 32-Learning Curve | GloVe Embeddings V7- Advanced Complexity

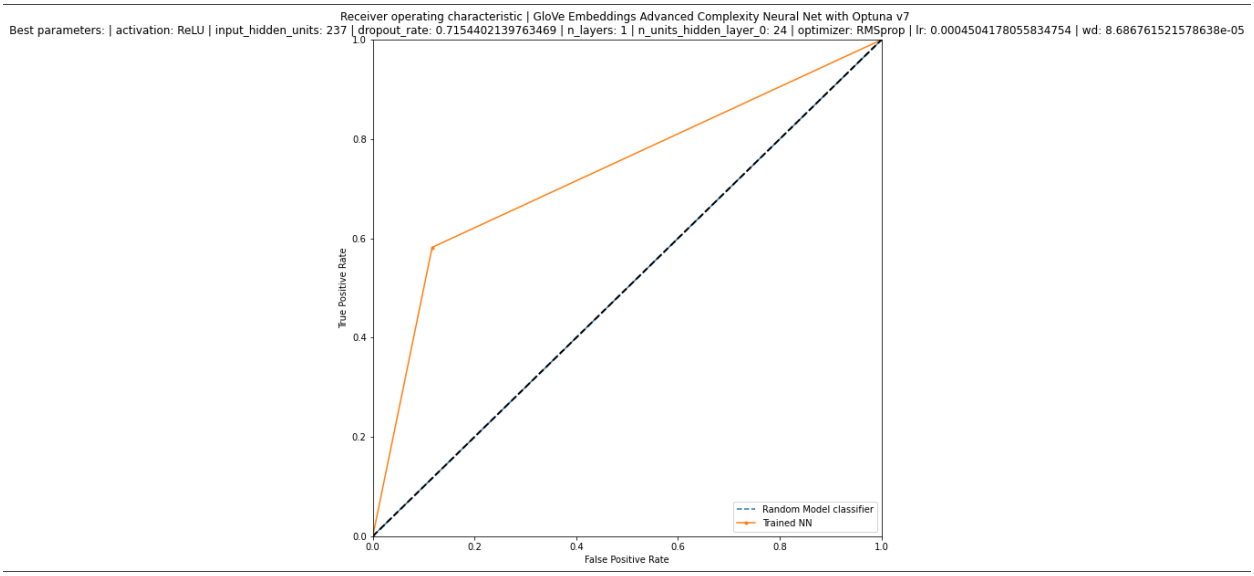


Figure 33-ROC Curve | GloVe Embeddings V7-Advanced Complexity

3.2.9 GloVe embeddings (three transformations) v8

For final experiment I have applied the following:

- 50 embeddings dimension for GloVe embeddings. Decreased the embeddings dimension from 200 to 50 dimensions.

The rest of the hyper-parameters are same as in the previous experiment (3.2.8).

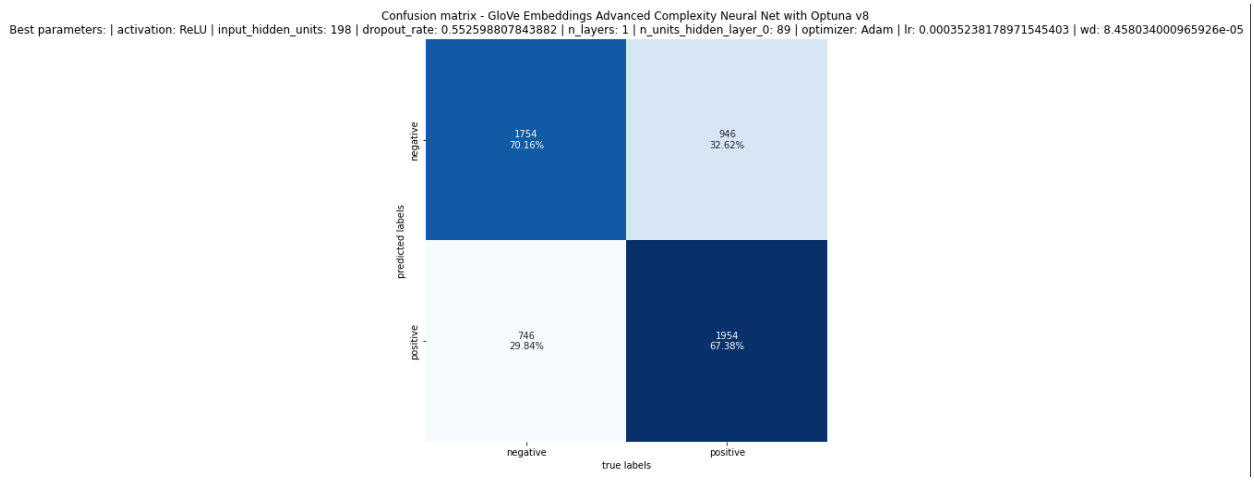


Figure 34--Evaluation Report | GloVe Embeddings V8-Advanced Complexity

FPR (False Positive Ratio): 29.84%

FNR (False Negative Ratio): 32.62%

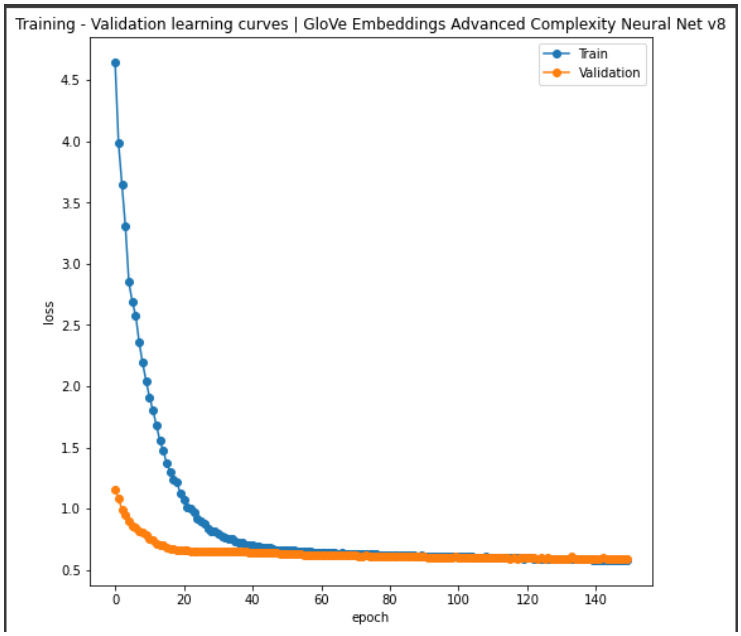


Figure 35-Learning Curve | GloVe Embeddings V8- Advanced Complexity

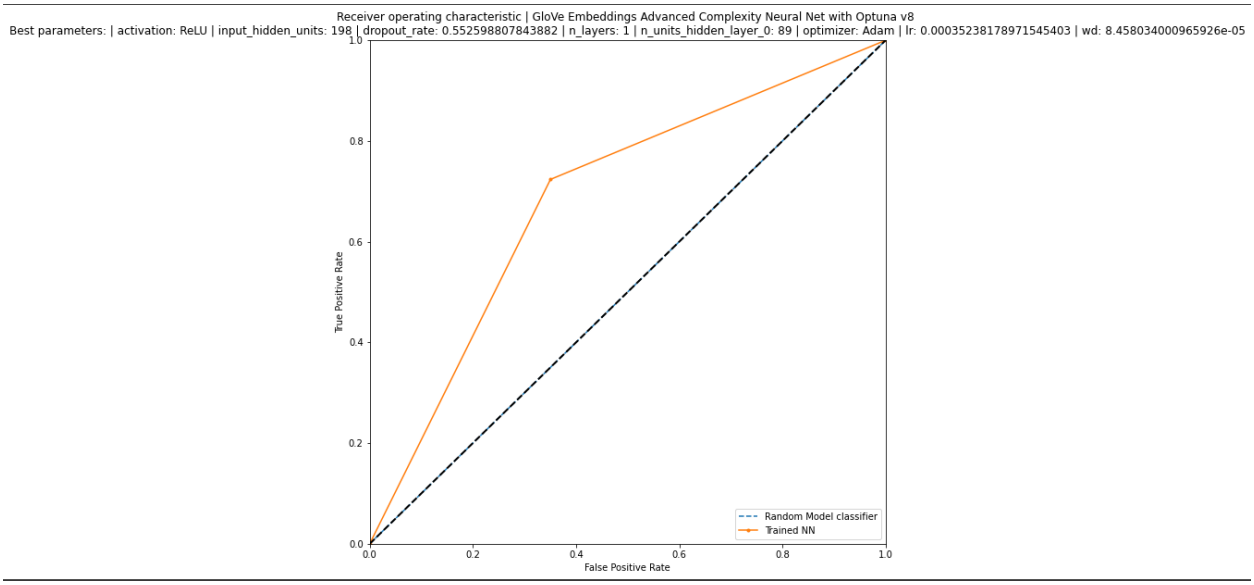


Figure 36-ROC Curve | GloVe Embeddings V8-Advanced Complexity

3.2.10 Evaluation metrics table

Below I report the evaluation metric results from the four advanced complexity experiments and I also comment of the results from all the previous figures to make a valid conclusion on the best trivial complexity model.

Model name	Precision score	Recall score	Roc score	F1-score	Bias	Variance
Trainable embeddings	0.8573	0.8678	0.8617	0.8625	0.0	0.25
GloVe embeddings	0.7145	0.5367	0.6611	0.6129	0.031	0.2345
GloVe embeddings (2)	0.6807	0.7533	0.7000	0.7152	0.0056	0.2472
GloVe embeddings (3)	0.7962	0.3056	0.6122	0.4407	0.188	0.156
GloVe embeddings (4)	0.6619	0.6633	0.6622	0.6626	0.0	0.25
GloVe embeddings (5)	0.6708	0.6837	0.6741	0.6772	0.000	0.2499
GloVe embeddings (6)	0.6589	0.9100	0.7194	0.7643	0.0726	0.2137
GloVe embeddings (7)	0.8330	0.5819	0.7326	0.6851	0.0454	0.2273
GloVe embeddings (8)	0.6738	0.7237	0.6867	0.6979	0.0028	0.2486

2-Evaluation metric results - Advanced complexity models

With green is marked the metric with the best performance over the three models. With orange is marked the GloVe model with the best performance from the other GloVe models.

3.3 Evaluation metrics table [Trivial & Advanced complexity models]

Experiment No.	Model name	Precision score	Recall score	Roc score	F1-score	Bias	Variance
Trivial Complexity							
1	Trainable embeddings	0.8498	0.8300	0.8417	0.8390	0.002	0.2498
2	GloVe embeddings	0.6624	0.6322	0.6550	0.6470	0.001	0.2495
3	GloVe embeddings (2)	0.7145	0.5367	0.6611	0.6129	0.031	0.2345
Advanced Complexity							
4	**Trainable embeddings**	0.8573	0.8678	0.8617	0.8625	0.0	0.25
5	GloVe embeddings	0.7145	0.5367	0.6611	0.6129	0.031	0.2345
6	GloVe embeddings (2)	0.6807	0.7533	0.7000	0.7152	0.0056	0.2472
78	GloVe embeddings (3)	0.7962	0.3056	0.6122	0.4407	0.188	0.156
8	GloVe embeddings (4)	0.6619	0.6633	0.6622	0.6626	0.0	0.25
9	GloVe embeddings (5)	0.6708	0.6837	0.6741	0.6772	0.000	0.2499
10	**GloVe embeddings (6)**	0.6589	0.9100	0.7194	0.7643	0.0726	0.2137
11	**GloVe embeddings (7)**	0.8330	0.5819	0.7326	0.6851	0.0454	0.2273
12	GloVe embeddings (8)	0.6738	0.7237	0.6867	0.6979	0.0028	0.2486

3-Evaluation metric results - Trivial & Advanced complexity models

With green is marked the metric with the best performance over the three models. With orange is marked the GloVe model with the best performance from the other GloVe models.

The clear winner of all the twelve (12) experiments is the neural network of the fourth experiment (marked in **bold)** trained on trainable word embeddings. Whereas from the ten (10) experiments with GloVe pre-trained embeddings the clear winner(s) are the eighth and ninth experiments (marked in **bold**) with the most orange highlighted scores.

The key outcomes from the experiments are reported below:

- Even though the three mathematical expressions did not improve the results, most of the experiments had better performance after the application of embedding transformation.
- The increase of hidden units, the increase of maximum length of vectors per sentence and the regularization applied with learning decay had positive impact on the models trained with GloVe embeddings.
- The application of Tanh() activation function on the weights of the embedding layer (seventh experiment) didn't not improve the prediction and learning results of the model classifier. The experiment only showed a decrease in variance but a notable increase in bias (underfit).

- The first three advanced complexity models trained with GloVe embeddings (sub-units 3.2.2, 3.2.3, 3.2.4) had a very flat training learning curve after 15 epochs. This could possibly mean that the model classifier fell into a local minimum on the validation results and couldn't observe any new data pattern.
- Using 165 words per sentence instead of 150 or 235 had better overall results (experiments 6,7,8).
- The increase of the training batch size from 64 to 128 and finally to 256 had a positive impact in the predictive power of the model (experiments 6-9).
- At least 100 epochs should be used for training.
- The GloVe embeddings with 200 dimensions produced the most accurate predictions.
- The learning rate adjustment and weight decays had a small impact on the predictive power of the model. It's worth testing the ability of the model to learn in low and high learning rates with or without L2 regularization (weight_decay parameter).

For the scoring of the grader's data sample (never-seen-before from the neural networks) I will send the best neural networks out of the twelve experiments. The best-performance model belongs to experiments: four and ten (check table 3).

4 Future work – Further improvements

1) Add more n-gram combinations. Like for example tri-grams, quad-grams in the custom vocabulary iterator object built for trainable embeddings.
2) Test other pre-trained embedding models, such as FastText embeddings.
3) Apply the changes of the sixth experiment [increase in hidden layer units, increase total words per sentence, add weight decay on optimizers] on the trivial and advanced complexity GloVe model without the application of the mathematical transformations on the embeddings layer.
4) Test a higher embedding dimension for the GloVe embeddings such as 300 embeddings matrix. A couple of trials were already executed; however, both runs failed due to lack of RAM in the Colab Notebook.
5) Import pre-trained embedding trained on a bigger vocabulary (i.e., 42B billion words). All the GloVe experiments were executed with 6B pre-trained embeddings except one experiment with 840B and 300 embeddings, which failed due to lack of available RAM.