University of Athens
Department of Informatics and Telecommunications

Deep Learning for NLP - HW2

Student name: *Dimitrios Chrysos*
sdi: 2100275

Course: *Artificial Intelligence II (ΥΣ19)*
Semester: *Spring Semester 2024-2025*

Contents

1. Abstract

The task is to perform sentiment analysis using PyTorch with Word2Vec word embeddings as input to the model. The project is written in Python for a given English-language Twitter dataset. Three datasets will be used: `train_dataset`, `val_dataset`, and `test_dataset`, which are used for training, validation, and testing, respectively.

2. Data processing and analysis

2.1. Pre-processing

The data cleaning and pre-processing were done using regular expressions and Python methods and applied to the three provided datasets. The following steps were used:

1. To ensure uniformity, all text is converted to lowercase
2. Common spelling mistakes, contractions, slang words, and informal abbreviations are corrected to improve standardization and prevent misinterpretations during feature extraction.
3. Removal of URLs, Hashtags, Emails, Mentions, Numbers, Emojis, Non-ASCII characters, Single-Letter Words, and Special Characters because they provide close to no value for sentiment analysis.
4. Sequences of three or more identical letters are reduced for standardization.
5. Excess whitespace is replaced with a single space for readability purposes.

Applying these techniques ensures that the text data is clean, structured, and suitable for sentiment analysis. This preprocessing step improves model accuracy by eliminating noise and standardizing input text.

2.2. Data partitioning for train, test and validation

- The data was already portioned.

2.3. Vectorization

- The technique uses the pre-trained Word2Vec model "GoogleNews-vectors-negative300" to map each word to a high-dimensional vector.
- A tweet is tokenized, and each token is replaced with its corresponding Google News vector (or a zero vector if absent).
- The total representation of the tweet is then calculated by averaging these word vectors, producing a semantically meaningful embedding of fixed size.

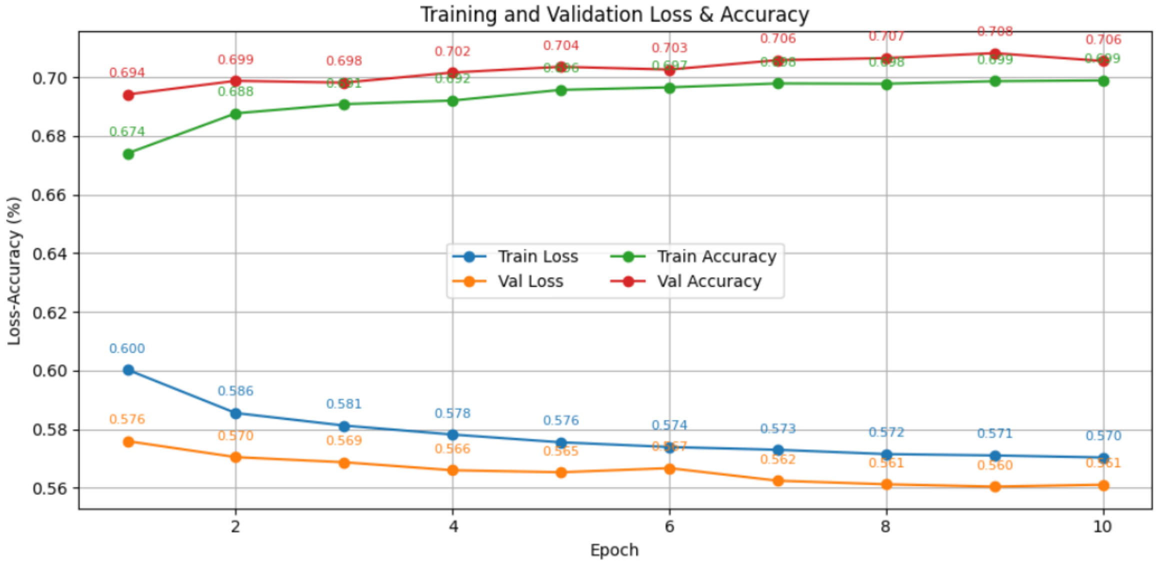
3. Algorithms and Experiments

3.1. Experiments

1. First Experiment - glove.6B.50d.txt

   - The first experiment used the following configuration:
     - Glove `glove.6B.50d.txt` for the embeddings.
     - `batch_size=128` for the data-loaders.
     - A model architecture of two hidden layers that both used **ReLu** as an activation function, **batch normalization** for regularization, and **dropout** to reduce overfitting.
     - `num_epochs=10` to get a starting understanding of the metrics.
     - `nn.CrossEntropyLoss()` for the loss function.
     - **Adam** optimizer.
     - The three variables that follow were generated using **optuna** optimization in previous discarded experiments and are the base for most of the following experiments.
       (a) `dropout_prob=0.3125125247127012`
       (b) `lr=0.0019524310388177159`
       (c) `weight_decay=5.263544573640884e-06`
   - This configuration produced the following results:

```
Epoch  1: Train Loss = 0.60029, Train Acc = 0.67400 | Val Loss = 0.57593, Val Acc = 0.69410
Epoch  2: Train Loss = 0.58550, Train Acc = 0.68767 | Val Loss = 0.57047, Val Acc = 0.69875
Epoch  3: Train Loss = 0.58123, Train Acc = 0.69080 | Val Loss = 0.56873, Val Acc = 0.69813
Epoch  4: Train Loss = 0.57818, Train Acc = 0.69203 | Val Loss = 0.56601, Val Acc = 0.70158
Epoch  5: Train Loss = 0.57552, Train Acc = 0.69567 | Val Loss = 0.56531, Val Acc = 0.70353
Epoch  6: Train Loss = 0.57392, Train Acc = 0.69653 | Val Loss = 0.56673, Val Acc = 0.70259
Epoch  7: Train Loss = 0.57298, Train Acc = 0.69785 | Val Loss = 0.56244, Val Acc = 0.70580
Epoch  8: Train Loss = 0.57151, Train Acc = 0.69776 | Val Loss = 0.56121, Val Acc = 0.70651
Epoch  9: Train Loss = 0.57105, Train Acc = 0.69862 | Val Loss = 0.56042, Val Acc = 0.70818
Epoch 10: Train Loss = 0.57035, Train Acc = 0.69889 | Val Loss = 0.56107, Val Acc = 0.70551
```



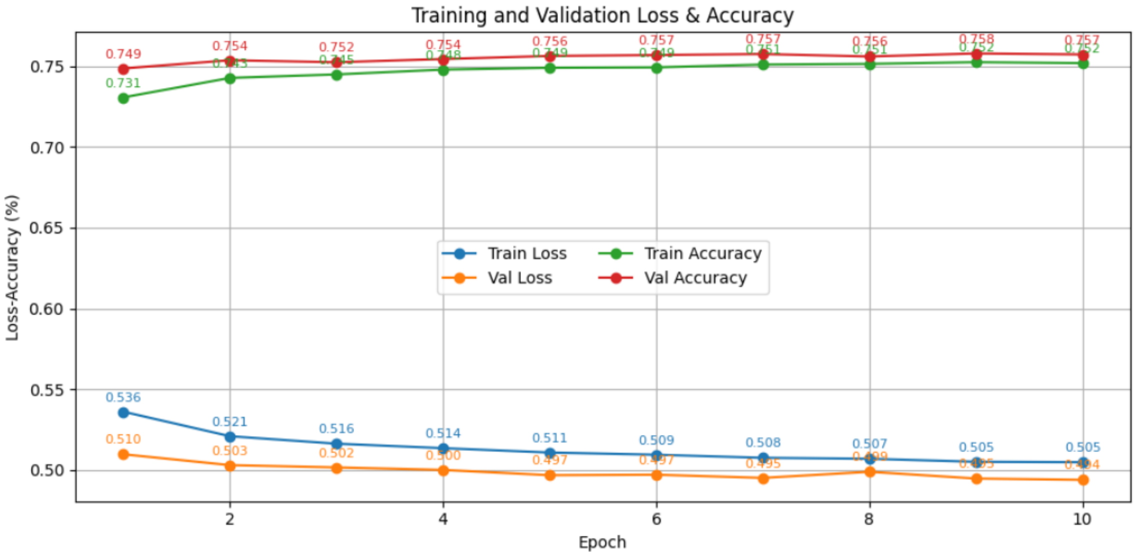

```
Accuracy: 0.705515
Classification Report:
              precision    recall  f1-score   support

           0       0.71      0.70      0.70     21197
           1       0.70      0.71      0.71     21199

    accuracy                           0.71     42396
   macro avg       0.71      0.71      0.71     42396
weighted avg       0.71      0.71      0.71     42396
```

Figure 1: Experiment 1

2. Second Experiment - Self Trained Vectors

   - For the second experiment, only the vectorization strategy changed.
     - The vectors used here are **trained** from the **training dataset**.
     - The rest of the configuration is the same as **Experiment 1**
     - This configuration produced the following results:

```
Epoch  1: Train Loss = 0.53610, Train Acc = 0.73061 | Val Loss = 0.50976, Val Acc = 0.74854
Epoch  2: Train Loss = 0.52095, Train Acc = 0.74267 | Val Loss = 0.50305, Val Acc = 0.75361
Epoch  3: Train Loss = 0.51632, Train Acc = 0.74482 | Val Loss = 0.50163, Val Acc = 0.75236
Epoch  4: Train Loss = 0.51355, Train Acc = 0.74782 | Val Loss = 0.50008, Val Acc = 0.75436
Epoch  5: Train Loss = 0.51083, Train Acc = 0.74896 | Val Loss = 0.49681, Val Acc = 0.75625
Epoch  6: Train Loss = 0.50948, Train Acc = 0.74920 | Val Loss = 0.49712, Val Acc = 0.75682
Epoch  7: Train Loss = 0.50758, Train Acc = 0.75094 | Val Loss = 0.49514, Val Acc = 0.75745
Epoch  8: Train Loss = 0.50697, Train Acc = 0.75137 | Val Loss = 0.49892, Val Acc = 0.75592
Epoch  9: Train Loss = 0.50511, Train Acc = 0.75241 | Val Loss = 0.49475, Val Acc = 0.75781
Epoch 10: Train Loss = 0.50486, Train Acc = 0.75182 | Val Loss = 0.49394, Val Acc = 0.75719
```




```
Accuracy: 0.757194
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.73      0.75     21197
           1       0.75      0.78      0.76     21199

    accuracy                           0.76     42396
   macro avg       0.76      0.76      0.76     42396
weighted avg       0.76      0.76      0.76     42396
```

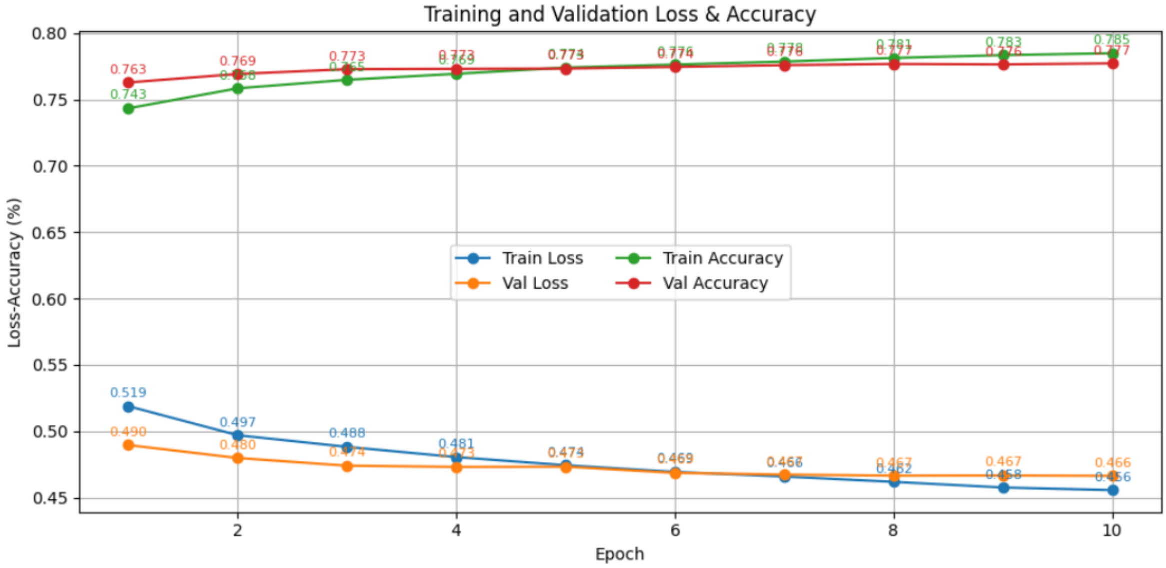
Figure 2: Experiment 2

- The accuracy of the evaluation dataset increased significantly.

3. Third Experiment - Google News Vectors

   - Following **Experiment 2**, the third experiment again changed only the vectorization strategy.
     - The vectors used here are from the `GoogleNews-vectors-negative300.bin` file.
     - The rest of the configuration is the same as **Experiment 1**
     - This configuration produced the following results:

```
Epoch  1: Train Loss = 0.51884, Train Acc = 0.74326 | Val Loss = 0.48955, Val Acc = 0.76274
Epoch  2: Train Loss = 0.49701, Train Acc = 0.75834 | Val Loss = 0.47983, Val Acc = 0.76913
Epoch  3: Train Loss = 0.48822, Train Acc = 0.76473 | Val Loss = 0.47404, Val Acc = 0.77276
Epoch  4: Train Loss = 0.48050, Train Acc = 0.76927 | Val Loss = 0.47311, Val Acc = 0.77300
Epoch  5: Train Loss = 0.47438, Train Acc = 0.77396 | Val Loss = 0.47327, Val Acc = 0.77321
Epoch  6: Train Loss = 0.46934, Train Acc = 0.77620 | Val Loss = 0.46861, Val Acc = 0.77444
Epoch  7: Train Loss = 0.46587, Train Acc = 0.77850 | Val Loss = 0.46732, Val Acc = 0.77580
Epoch  8: Train Loss = 0.46185, Train Acc = 0.78121 | Val Loss = 0.46654, Val Acc = 0.77670
Epoch  9: Train Loss = 0.45761, Train Acc = 0.78331 | Val Loss = 0.46666, Val Acc = 0.77635
Epoch 10: Train Loss = 0.45564, Train Acc = 0.78477 | Val Loss = 0.46638, Val Acc = 0.77715
```




```
Accuracy: 0.777149
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.80      0.78     21197
           1       0.79      0.75      0.77     21199

    accuracy                           0.78     42396
   macro avg       0.78      0.78      0.78     42396
weighted avg       0.78      0.78      0.78     42396
```

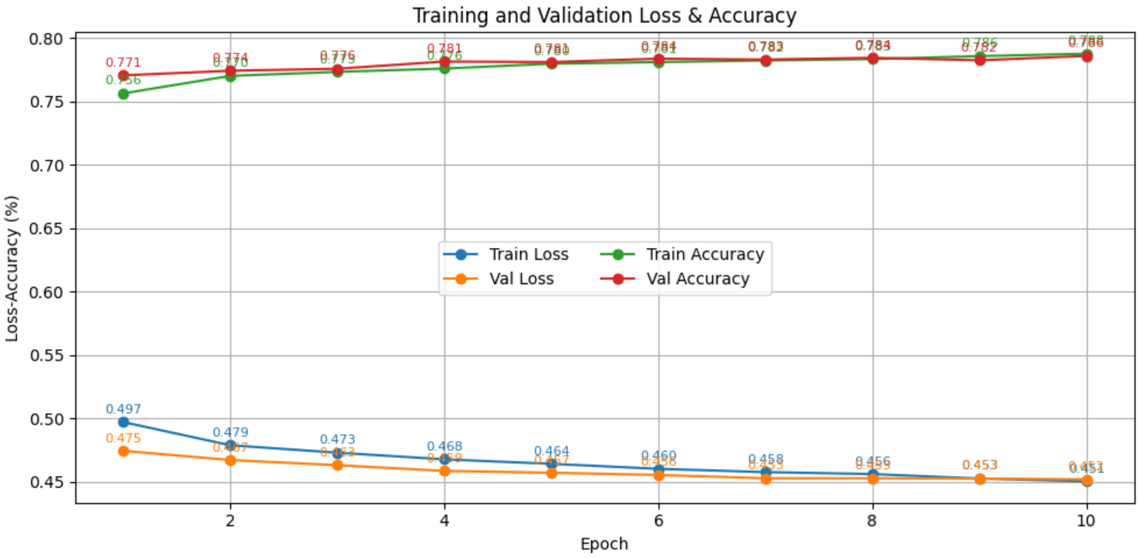
Figure 3: Experiment 3

- The accuracy of the evaluation dataset again increased significantly.

4. Fourth Experiment - glove.twitter.27B.200d.txt

   - Again following the two previous experiments, the fourth experiment changed only the vectorization strategy.
     - The vectors used here are from the `glove.twitter.27b.200d.txt` file.
     - The rest of the configuration is the same as **Experiment 1**
     - This configuration produced the following results:

```
Epoch  1: Train Loss = 0.49724, Train Acc = 0.75641 | Val Loss = 0.47461, Val Acc = 0.77062
Epoch  2: Train Loss = 0.47901, Train Acc = 0.77019 | Val Loss = 0.46736, Val Acc = 0.77422
Epoch  3: Train Loss = 0.47314, Train Acc = 0.77334 | Val Loss = 0.46332, Val Acc = 0.77583
Epoch  4: Train Loss = 0.46791, Train Acc = 0.77597 | Val Loss = 0.45882, Val Acc = 0.78149
Epoch  5: Train Loss = 0.46435, Train Acc = 0.77979 | Val Loss = 0.45732, Val Acc = 0.78109
Epoch  6: Train Loss = 0.46043, Train Acc = 0.78105 | Val Loss = 0.45558, Val Acc = 0.78375
Epoch  7: Train Loss = 0.45785, Train Acc = 0.78233 | Val Loss = 0.45299, Val Acc = 0.78305
Epoch  8: Train Loss = 0.45630, Train Acc = 0.78342 | Val Loss = 0.45298, Val Acc = 0.78444
Epoch  9: Train Loss = 0.45279, Train Acc = 0.78575 | Val Loss = 0.45279, Val Acc = 0.78246
Epoch 10: Train Loss = 0.45058, Train Acc = 0.78768 | Val Loss = 0.45211, Val Acc = 0.78590
```




```
Accuracy: 0.785900
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.81      0.79     21197
           1       0.80      0.77      0.78     21199

    accuracy                           0.79     42396
   macro avg       0.79      0.79      0.79     42396
weighted avg       0.79      0.79      0.79     42396
```

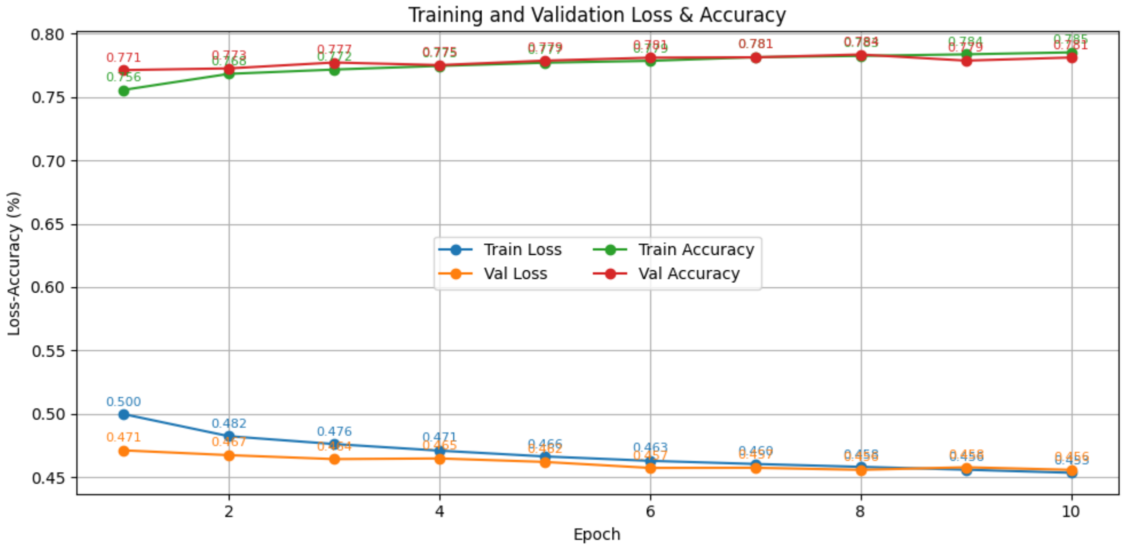
Figure 4: Experiment 4

- The accuracy of the evaluation dataset again increased significantly.

5. Fifth Experiment - Try removing most frequent words

   - From this experiment and the experiments moving forward, the model from **Experiment 4** will be used.
   - The purpose of this experiment is to discover whether the removal of words that are included in 15% (or more) of the tweets increases the accuracy of the evaluation dataset.
   - The rest of the configuration is the same as **Experiment 4**
   - This change in the preprocessing step produced the following results:

```
Epoch  1: Train Loss = 0.49972, Train Acc = 0.75570 | Val Loss = 0.47113, Val Acc = 0.77132
Epoch  2: Train Loss = 0.48231, Train Acc = 0.76835 | Val Loss = 0.46736, Val Acc = 0.77267
Epoch  3: Train Loss = 0.47612, Train Acc = 0.77175 | Val Loss = 0.46424, Val Acc = 0.77731
Epoch  4: Train Loss = 0.47093, Train Acc = 0.77463 | Val Loss = 0.46476, Val Acc = 0.77526
Epoch  5: Train Loss = 0.46634, Train Acc = 0.77716 | Val Loss = 0.46200, Val Acc = 0.77878
Epoch  6: Train Loss = 0.46288, Train Acc = 0.77861 | Val Loss = 0.45729, Val Acc = 0.78111
Epoch  7: Train Loss = 0.46040, Train Acc = 0.78150 | Val Loss = 0.45739, Val Acc = 0.78142
Epoch  8: Train Loss = 0.45814, Train Acc = 0.78256 | Val Loss = 0.45577, Val Acc = 0.78364
Epoch  9: Train Loss = 0.45588, Train Acc = 0.78376 | Val Loss = 0.45780, Val Acc = 0.77880
Epoch 10: Train Loss = 0.45345, Train Acc = 0.78530 | Val Loss = 0.45578, Val Acc = 0.78125
```




```
Accuracy: 0.781253
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.77      0.78     21197
           1       0.77      0.80      0.78     21199

    accuracy                           0.78     42396
   macro avg       0.78      0.78      0.78     42396
weighted avg       0.78      0.78      0.78     42396
```

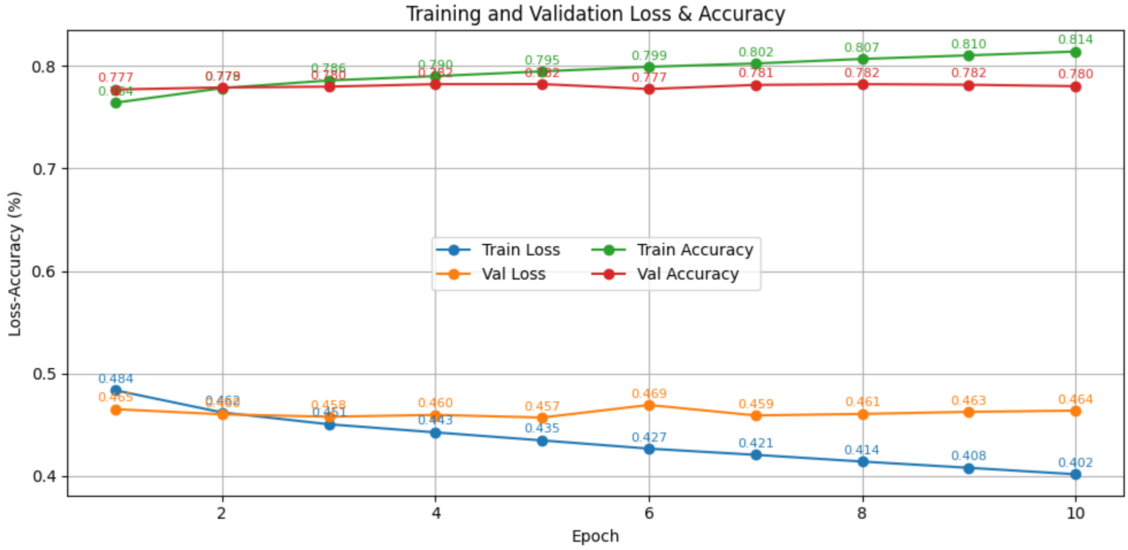
Figure 5: Experiment 5

- The metrics across the board got worse compared to **Experiment 4**. For this reason, this technique will not be used in the next experiments.

6. Sixth Experiment - Try not using dropout

   - The purpose of this experiment was to understand the impact of **dropout**.
   - For this reason the experiment runs with `dropout_prob=0`
   - The rest of the configuration is the same as **Experiment 4**
   - The following are the results:

```
Epoch  1: Train Loss = 0.48369, Train Acc = 0.76415 | Val Loss = 0.46531, Val Acc = 0.77701
Epoch  2: Train Loss = 0.46212, Train Acc = 0.77834 | Val Loss = 0.46027, Val Acc = 0.77901
Epoch  3: Train Loss = 0.45058, Train Acc = 0.78585 | Val Loss = 0.45792, Val Acc = 0.77981
Epoch  4: Train Loss = 0.44267, Train Acc = 0.79007 | Val Loss = 0.45972, Val Acc = 0.78222
Epoch  5: Train Loss = 0.43484, Train Acc = 0.79469 | Val Loss = 0.45717, Val Acc = 0.78229
Epoch  6: Train Loss = 0.42684, Train Acc = 0.79915 | Val Loss = 0.46941, Val Acc = 0.77748
Epoch  7: Train Loss = 0.42077, Train Acc = 0.80239 | Val Loss = 0.45919, Val Acc = 0.78149
Epoch  8: Train Loss = 0.41423, Train Acc = 0.80681 | Val Loss = 0.46067, Val Acc = 0.78222
Epoch  9: Train Loss = 0.40812, Train Acc = 0.81031 | Val Loss = 0.46269, Val Acc = 0.78165
Epoch 10: Train Loss = 0.40183, Train Acc = 0.81406 | Val Loss = 0.46388, Val Acc = 0.78024
```




```
Accuracy: 0.780239
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.82      0.79     21197
           1       0.80      0.74      0.77     21199

    accuracy                           0.78     42396
   macro avg       0.78      0.78      0.78     42396
weighted avg       0.78      0.78      0.78     42396
```

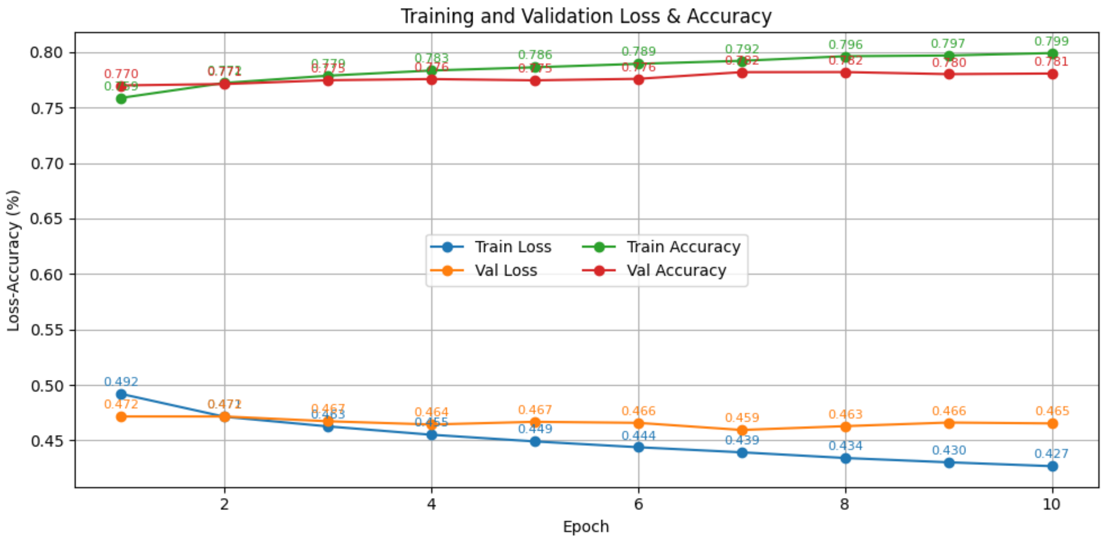
Figure 6: Experiment 6

- The results clearly show both a decrease in accuracy and also a significant increase in overfitting that can be spotted by looking at the differences between the curves `Train Accuracy-Val Accuracy` and `Train Loss-Val Loss`.
- For the above reason, the next experiments will use dropout.
- **Experiment 4** is the best so far.

7. Seventh Experiment - Try using an 1 hidden layer architecture

   - For this experiment, the network architecture changes from two hidden layers to one.
   - The rest of the configuration is the same as **Experiment 4**
   - This change in the architecture produced the following results:

```
Experiment 7 - Started
Epoch  1: Train Loss = 0.49189, Train Acc = 0.75875 | Val Loss = 0.47154, Val Acc = 0.76998
Epoch  2: Train Loss = 0.47127, Train Acc = 0.77221 | Val Loss = 0.47154, Val Acc = 0.77123
Epoch  3: Train Loss = 0.46259, Train Acc = 0.77875 | Val Loss = 0.46723, Val Acc = 0.77467
Epoch  4: Train Loss = 0.45507, Train Acc = 0.78340 | Val Loss = 0.46431, Val Acc = 0.77571
Epoch  5: Train Loss = 0.44904, Train Acc = 0.78631 | Val Loss = 0.46673, Val Acc = 0.77458
Epoch  6: Train Loss = 0.44382, Train Acc = 0.78944 | Val Loss = 0.46580, Val Acc = 0.77592
Epoch  7: Train Loss = 0.43923, Train Acc = 0.79217 | Val Loss = 0.45938, Val Acc = 0.78194
Epoch  8: Train Loss = 0.43412, Train Acc = 0.79628 | Val Loss = 0.46282, Val Acc = 0.78201
Epoch  9: Train Loss = 0.43024, Train Acc = 0.79697 | Val Loss = 0.46608, Val Acc = 0.78014
Epoch 10: Train Loss = 0.42676, Train Acc = 0.79915 | Val Loss = 0.46529, Val Acc = 0.78071
```




```
Accuracy: 0.780710
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.77      0.78     21197
           1       0.78      0.79      0.78     21199

    accuracy                           0.78     42396
   macro avg       0.78      0.78      0.78     42396
weighted avg       0.78      0.78      0.78     42396
```

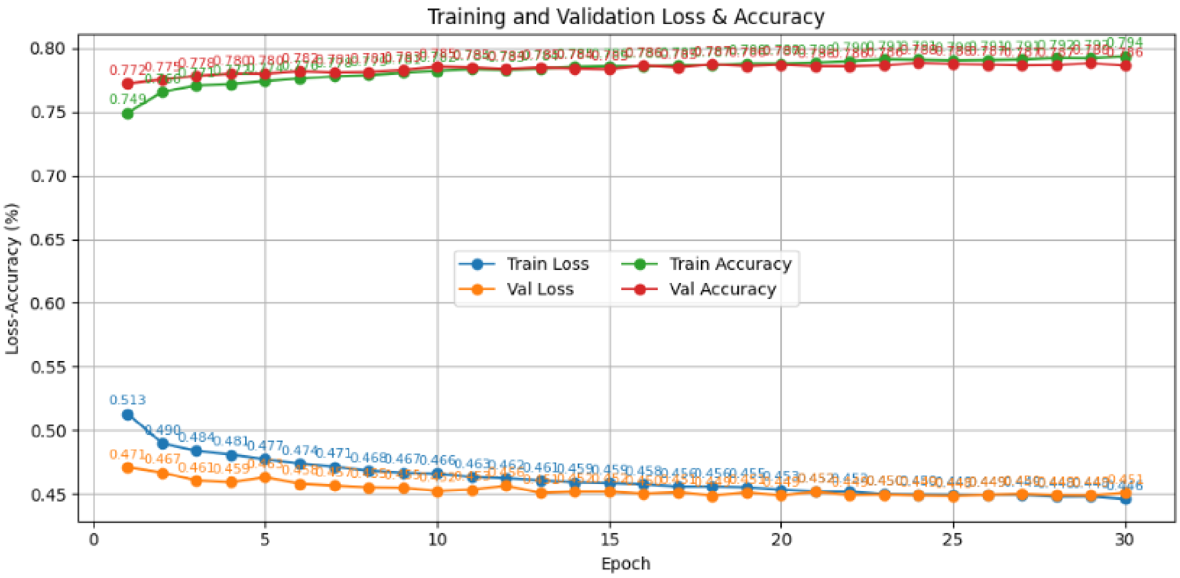
Figure 7: Experiment 7

- Although the difference is smaller, we are having the same problem with overfitting the **Experiment 6** had.
- The classification report metrics are also decreased compared to **Experiment 4**.
- **Experiment 4** is still the best performing.

8. Eighth Experiment - Try using a 3 hidden layers architecture

    - In contrast with the previous experiment, this one uses more hidden layers.
    - The architecture changes from two hidden layers to three.
    - Other configuration changes are `num_epochs=30` and `dropout_prob=0.4125125247127012` (to regularize the model), those changes are needed to understand if this more complex model is indeed better than the one of **Experiment 4**.
    - Results:

```
Experiment 8 - Started
Epoch  1: Train Loss = 0.51299, Train Acc = 0.74908 | Val Loss = 0.47097, Val Acc = 0.77210
Epoch  2: Train Loss = 0.48984, Train Acc = 0.76561 | Val Loss = 0.46672, Val Acc = 0.77545
Epoch  3: Train Loss = 0.48379, Train Acc = 0.77078 | Val Loss = 0.46055, Val Acc = 0.77823
Epoch  4: Train Loss = 0.48086, Train Acc = 0.77183 | Val Loss = 0.45916, Val Acc = 0.78000
Epoch  5: Train Loss = 0.47728, Train Acc = 0.77408 | Val Loss = 0.46300, Val Acc = 0.78000
Epoch  6: Train Loss = 0.47379, Train Acc = 0.77619 | Val Loss = 0.45809, Val Acc = 0.78189
Epoch  7: Train Loss = 0.47133, Train Acc = 0.77777 | Val Loss = 0.45656, Val Acc = 0.78088
Epoch  8: Train Loss = 0.46829, Train Acc = 0.77872 | Val Loss = 0.45521, Val Acc = 0.78128
Epoch  9: Train Loss = 0.46667, Train Acc = 0.78065 | Val Loss = 0.45488, Val Acc = 0.78286
Epoch 10: Train Loss = 0.46583, Train Acc = 0.78225 | Val Loss = 0.45232, Val Acc = 0.78538
Epoch 11: Train Loss = 0.46334, Train Acc = 0.78350 | Val Loss = 0.45344, Val Acc = 0.78479
Epoch 12: Train Loss = 0.46233, Train Acc = 0.78275 | Val Loss = 0.45644, Val Acc = 0.78354
Epoch 13: Train Loss = 0.46055, Train Acc = 0.78392 | Val Loss = 0.45092, Val Acc = 0.78479
Epoch 14: Train Loss = 0.45912, Train Acc = 0.78529 | Val Loss = 0.45182, Val Acc = 0.78401
Epoch 15: Train Loss = 0.45854, Train Acc = 0.78582 | Val Loss = 0.45188, Val Acc = 0.78340
Epoch 16: Train Loss = 0.45763, Train Acc = 0.78561 | Val Loss = 0.45015, Val Acc = 0.78642
Epoch 17: Train Loss = 0.45583, Train Acc = 0.78669 | Val Loss = 0.45132, Val Acc = 0.78491
Epoch 18: Train Loss = 0.45591, Train Acc = 0.78666 | Val Loss = 0.44870, Val Acc = 0.78731
Epoch 19: Train Loss = 0.45524, Train Acc = 0.78804 | Val Loss = 0.45114, Val Acc = 0.78592
Epoch 20: Train Loss = 0.45335, Train Acc = 0.78799 | Val Loss = 0.44879, Val Acc = 0.78703
Epoch 21: Train Loss = 0.45199, Train Acc = 0.78870 | Val Loss = 0.45163, Val Acc = 0.78585
Epoch 22: Train Loss = 0.45182, Train Acc = 0.78988 | Val Loss = 0.44914, Val Acc = 0.78581
Epoch 23: Train Loss = 0.44981, Train Acc = 0.79126 | Val Loss = 0.44957, Val Acc = 0.78649
Epoch 24: Train Loss = 0.44959, Train Acc = 0.79095 | Val Loss = 0.44881, Val Acc = 0.78835
Epoch 25: Train Loss = 0.44937, Train Acc = 0.79039 | Val Loss = 0.44846, Val Acc = 0.78750
Epoch 26: Train Loss = 0.44919, Train Acc = 0.79074 | Val Loss = 0.44933, Val Acc = 0.78710
Epoch 27: Train Loss = 0.44932, Train Acc = 0.79115 | Val Loss = 0.44998, Val Acc = 0.78651
Epoch 28: Train Loss = 0.44790, Train Acc = 0.79233 | Val Loss = 0.44904, Val Acc = 0.78675
Epoch 29: Train Loss = 0.44808, Train Acc = 0.79215 | Val Loss = 0.44883, Val Acc = 0.78814
Epoch 30: Train Loss = 0.44608, Train Acc = 0.79353 | Val Loss = 0.45075, Val Acc = 0.78637
```




```
Accuracy: 0.786371
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.79      0.79     21197
           1       0.79      0.78      0.79     21199

    accuracy                           0.79     42396
   macro avg       0.79      0.79      0.79     42396
weighted avg       0.79      0.79      0.79     42396
```

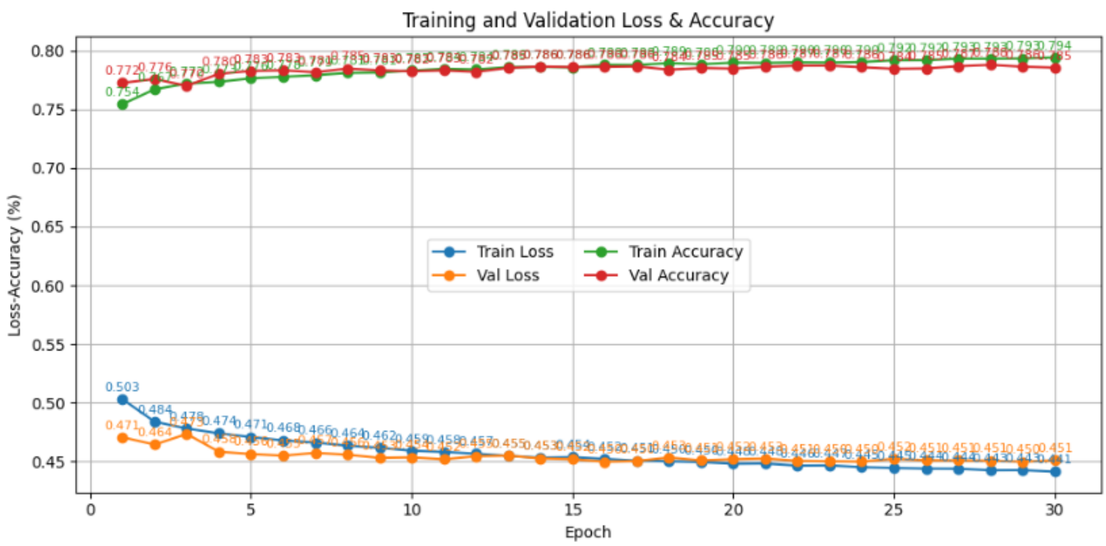
Figure 8: Experiment 8

- The results indicate better performance on the curves and more stable metrics in the classification report. To have a confident answer, the next experiment is the model with the architecture of **Experiment 4** but with the configuration changes of this experiment.

9. Ninth Experiment - 2 hidden layers with more `epochs` and `dropout_prob`

   - This experiment will prove if the previous one is indeed better than **Experiment 4**.
   - We will understand this by having the same architecture as **Experiment 4** but with the configuration changes of **Experiment 8**.
   - Results:

```
Epoch  1: Train Loss = 0.50307, Train Acc = 0.75427 | Val Loss = 0.47067, Val Acc = 0.77238
Epoch  2: Train Loss = 0.48390, Train Acc = 0.76693 | Val Loss = 0.46442, Val Acc = 0.77566
Epoch  3: Train Loss = 0.47812, Train Acc = 0.77179 | Val Loss = 0.47346, Val Acc = 0.76991
Epoch  4: Train Loss = 0.47399, Train Acc = 0.77330 | Val Loss = 0.45833, Val Acc = 0.78012
Epoch  5: Train Loss = 0.47084, Train Acc = 0.77626 | Val Loss = 0.45640, Val Acc = 0.78257
Epoch  6: Train Loss = 0.46799, Train Acc = 0.77760 | Val Loss = 0.45497, Val Acc = 0.78307
Epoch  7: Train Loss = 0.46614, Train Acc = 0.77892 | Val Loss = 0.45741, Val Acc = 0.78149
Epoch  8: Train Loss = 0.46355, Train Acc = 0.78089 | Val Loss = 0.45577, Val Acc = 0.78458
Epoch  9: Train Loss = 0.46171, Train Acc = 0.78116 | Val Loss = 0.45313, Val Acc = 0.78307
Epoch 10: Train Loss = 0.45921, Train Acc = 0.78288 | Val Loss = 0.45358, Val Acc = 0.78224
Epoch 11: Train Loss = 0.45820, Train Acc = 0.78435 | Val Loss = 0.45191, Val Acc = 0.78300
Epoch 12: Train Loss = 0.45665, Train Acc = 0.78383 | Val Loss = 0.45454, Val Acc = 0.78175
Epoch 13: Train Loss = 0.45487, Train Acc = 0.78571 | Val Loss = 0.45502, Val Acc = 0.78514
Epoch 14: Train Loss = 0.45326, Train Acc = 0.78607 | Val Loss = 0.45267, Val Acc = 0.78632
Epoch 15: Train Loss = 0.45373, Train Acc = 0.78563 | Val Loss = 0.45198, Val Acc = 0.78578
Epoch 16: Train Loss = 0.45224, Train Acc = 0.78779 | Val Loss = 0.44982, Val Acc = 0.78606
Epoch 17: Train Loss = 0.45071, Train Acc = 0.78762 | Val Loss = 0.45046, Val Acc = 0.78642
Epoch 18: Train Loss = 0.45023, Train Acc = 0.78927 | Val Loss = 0.45338, Val Acc = 0.78366
Epoch 19: Train Loss = 0.44953, Train Acc = 0.78828 | Val Loss = 0.45093, Val Acc = 0.78505
Epoch 20: Train Loss = 0.44808, Train Acc = 0.78963 | Val Loss = 0.45198, Val Acc = 0.78456
Epoch 21: Train Loss = 0.44830, Train Acc = 0.78940 | Val Loss = 0.45256, Val Acc = 0.78611
Epoch 22: Train Loss = 0.44649, Train Acc = 0.78989 | Val Loss = 0.45057, Val Acc = 0.78706
Epoch 23: Train Loss = 0.44668, Train Acc = 0.78969 | Val Loss = 0.45024, Val Acc = 0.78710
Epoch 24: Train Loss = 0.44521, Train Acc = 0.79017 | Val Loss = 0.44993, Val Acc = 0.78571
Epoch 25: Train Loss = 0.44453, Train Acc = 0.79167 | Val Loss = 0.45229, Val Acc = 0.78439
Epoch 26: Train Loss = 0.44399, Train Acc = 0.79178 | Val Loss = 0.45105, Val Acc = 0.78470
Epoch 27: Train Loss = 0.44389, Train Acc = 0.79300 | Val Loss = 0.45087, Val Acc = 0.78661
Epoch 28: Train Loss = 0.44257, Train Acc = 0.79295 | Val Loss = 0.45065, Val Acc = 0.78774
Epoch 29: Train Loss = 0.44267, Train Acc = 0.79319 | Val Loss = 0.44986, Val Acc = 0.78618
Epoch 30: Train Loss = 0.44140, Train Acc = 0.79410 | Val Loss = 0.45088, Val Acc = 0.78538
```




```
Accuracy: 0.785381
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.79      0.79     21197
           1       0.79      0.79      0.79     21199

    accuracy                           0.79     42396
   macro avg       0.79      0.79      0.79     42396
weighted avg       0.79      0.79      0.79     42396
```

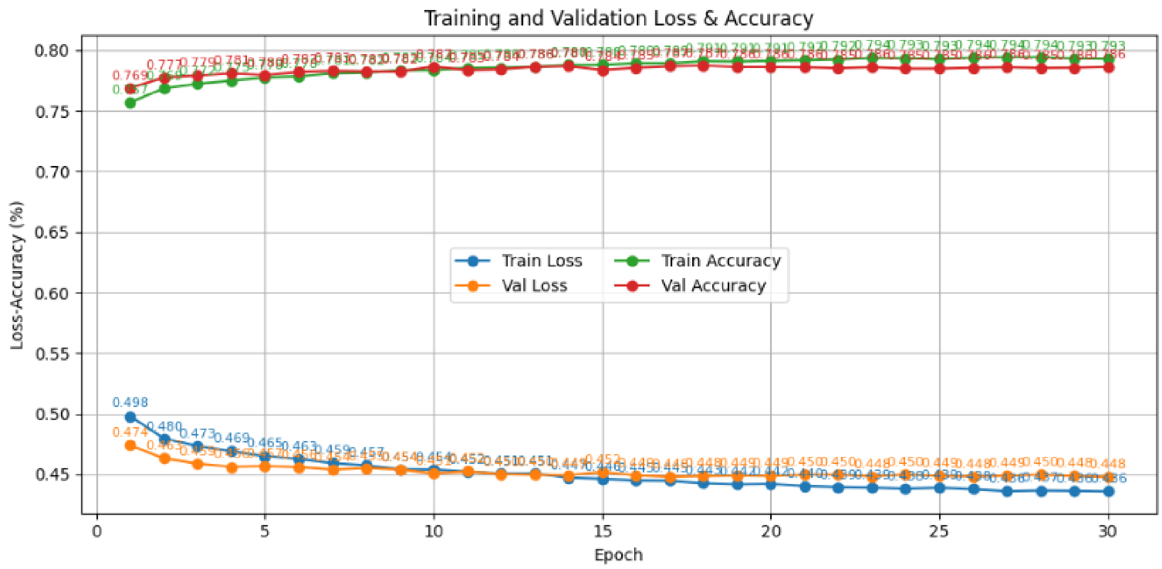
Figure 9: Experiment 9

- The results are almost identical between the two experiments.
- For this reason and because it has a lower complexity **Experiment 9** is considered the best.

10. Tenth Experiment - Try using GELU with 2 layers

    - The parameter that changes here is the activation function used, now the two hidden layers use the **GELU** activation function.
    - Results:

```
Epoch  1: Train Loss = 0.49805, Train Acc = 0.75702 | Val Loss = 0.47385, Val Acc = 0.76896
Epoch  2: Train Loss = 0.47951, Train Acc = 0.76851 | Val Loss = 0.46318, Val Acc = 0.77746
Epoch  3: Train Loss = 0.47335, Train Acc = 0.77236 | Val Loss = 0.45879, Val Acc = 0.77906
Epoch  4: Train Loss = 0.46906, Train Acc = 0.77495 | Val Loss = 0.45615, Val Acc = 0.78092
Epoch  5: Train Loss = 0.46498, Train Acc = 0.77750 | Val Loss = 0.45687, Val Acc = 0.77951
Epoch  6: Train Loss = 0.46269, Train Acc = 0.77837 | Val Loss = 0.45596, Val Acc = 0.78210
Epoch  7: Train Loss = 0.45924, Train Acc = 0.78091 | Val Loss = 0.45387, Val Acc = 0.78300
Epoch  8: Train Loss = 0.45740, Train Acc = 0.78163 | Val Loss = 0.45511, Val Acc = 0.78255
Epoch  9: Train Loss = 0.45410, Train Acc = 0.78337 | Val Loss = 0.45396, Val Acc = 0.78236
Epoch 10: Train Loss = 0.45388, Train Acc = 0.78367 | Val Loss = 0.45068, Val Acc = 0.78658
Epoch 11: Train Loss = 0.45208, Train Acc = 0.78529 | Val Loss = 0.45236, Val Acc = 0.78349
Epoch 12: Train Loss = 0.45059, Train Acc = 0.78568 | Val Loss = 0.45008, Val Acc = 0.78427
Epoch 13: Train Loss = 0.45066, Train Acc = 0.78647 | Val Loss = 0.44951, Val Acc = 0.78637
Epoch 14: Train Loss = 0.44717, Train Acc = 0.78782 | Val Loss = 0.44926, Val Acc = 0.78687
Epoch 15: Train Loss = 0.44621, Train Acc = 0.78791 | Val Loss = 0.45190, Val Acc = 0.78368
Epoch 16: Train Loss = 0.44492, Train Acc = 0.78930 | Val Loss = 0.44910, Val Acc = 0.78550
Epoch 17: Train Loss = 0.44473, Train Acc = 0.78923 | Val Loss = 0.44791, Val Acc = 0.78673
Epoch 18: Train Loss = 0.44261, Train Acc = 0.79093 | Val Loss = 0.44849, Val Acc = 0.78734
Epoch 19: Train Loss = 0.44170, Train Acc = 0.79066 | Val Loss = 0.44893, Val Acc = 0.78628
Epoch 20: Train Loss = 0.44210, Train Acc = 0.79128 | Val Loss = 0.44862, Val Acc = 0.78623
Epoch 21: Train Loss = 0.44038, Train Acc = 0.79178 | Val Loss = 0.44963, Val Acc = 0.78597
Epoch 22: Train Loss = 0.43938, Train Acc = 0.79246 | Val Loss = 0.44976, Val Acc = 0.78514
Epoch 23: Train Loss = 0.43898, Train Acc = 0.79357 | Val Loss = 0.44830, Val Acc = 0.78602
Epoch 24: Train Loss = 0.43815, Train Acc = 0.79343 | Val Loss = 0.44980, Val Acc = 0.78479
Epoch 25: Train Loss = 0.43894, Train Acc = 0.79268 | Val Loss = 0.44861, Val Acc = 0.78481
Epoch 26: Train Loss = 0.43779, Train Acc = 0.79391 | Val Loss = 0.44816, Val Acc = 0.78552
Epoch 27: Train Loss = 0.43613, Train Acc = 0.79425 | Val Loss = 0.44880, Val Acc = 0.78597
Epoch 28: Train Loss = 0.43668, Train Acc = 0.79431 | Val Loss = 0.44969, Val Acc = 0.78536
Epoch 29: Train Loss = 0.43641, Train Acc = 0.79326 | Val Loss = 0.44846, Val Acc = 0.78557
Epoch 30: Train Loss = 0.43592, Train Acc = 0.79293 | Val Loss = 0.44763, Val Acc = 0.78632
```




```
Accuracy: 0.786324
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.80      0.79     21197
           1       0.79      0.78      0.78     21199

    accuracy                           0.79     42396
   macro avg       0.79      0.79      0.79     42396
weighted avg       0.79      0.79      0.79     42396
```

Figure 10: Experiment 10

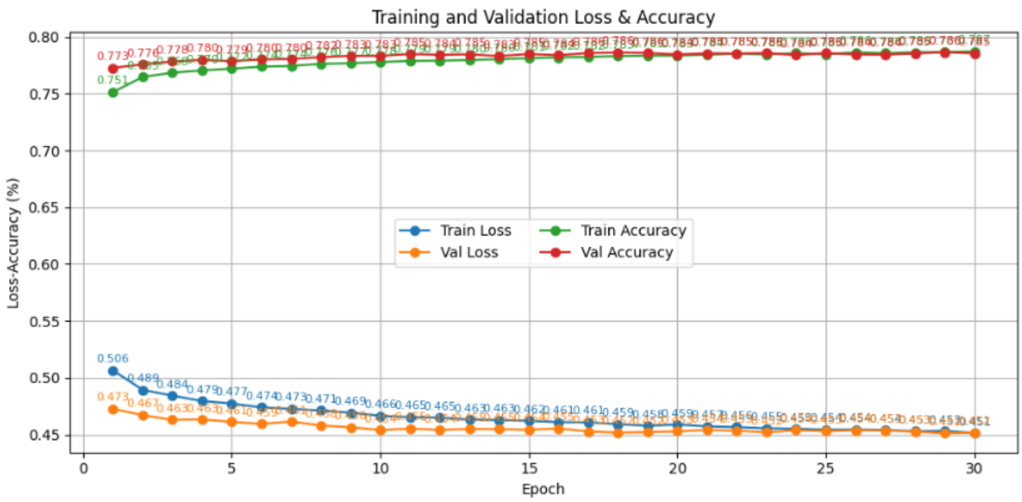
- Similar results with **Experiment 9**, but with a bit less stable metrics in the classification report. `ReLu` is also considered a more efficient approach for shallow networks.

11. Eleventh Experiment - Reduce Batch Size

- This experiment uses the same configuration as **Experiment 9** but with `batch_size=64` instead of `batch_size=128`.

- Results:

```
Epoch  1: Train Loss = 0.50634, Train Acc = 0.75133 | Val Loss = 0.47259, Val Acc = 0.77269
Epoch  2: Train Loss = 0.48930, Train Acc = 0.76468 | Val Loss = 0.46699, Val Acc = 0.77566
Epoch  3: Train Loss = 0.48415, Train Acc = 0.76842 | Val Loss = 0.46292, Val Acc = 0.77828
Epoch  4: Train Loss = 0.47949, Train Acc = 0.77042 | Val Loss = 0.46325, Val Acc = 0.77965
Epoch  5: Train Loss = 0.47723, Train Acc = 0.77193 | Val Loss = 0.46112, Val Acc = 0.77854
Epoch  6: Train Loss = 0.47379, Train Acc = 0.77374 | Val Loss = 0.45928, Val Acc = 0.78010
Epoch  7: Train Loss = 0.47254, Train Acc = 0.77437 | Val Loss = 0.46142, Val Acc = 0.78047
Epoch  8: Train Loss = 0.47109, Train Acc = 0.77616 | Val Loss = 0.45786, Val Acc = 0.78205
Epoch  9: Train Loss = 0.46917, Train Acc = 0.77675 | Val Loss = 0.45636, Val Acc = 0.78326
Epoch 10: Train Loss = 0.46646, Train Acc = 0.77777 | Val Loss = 0.45395, Val Acc = 0.78323
Epoch 11: Train Loss = 0.46515, Train Acc = 0.77881 | Val Loss = 0.45515, Val Acc = 0.78484
Epoch 12: Train Loss = 0.46494, Train Acc = 0.77902 | Val Loss = 0.45410, Val Acc = 0.78408
Epoch 13: Train Loss = 0.46321, Train Acc = 0.77954 | Val Loss = 0.45486, Val Acc = 0.78453
Epoch 14: Train Loss = 0.46268, Train Acc = 0.78034 | Val Loss = 0.45470, Val Acc = 0.78307
Epoch 15: Train Loss = 0.46200, Train Acc = 0.78119 | Val Loss = 0.45417, Val Acc = 0.78481
Epoch 16: Train Loss = 0.46087, Train Acc = 0.78182 | Val Loss = 0.45535, Val Acc = 0.78380
Epoch 17: Train Loss = 0.46065, Train Acc = 0.78227 | Val Loss = 0.45271, Val Acc = 0.78571
Epoch 18: Train Loss = 0.45905, Train Acc = 0.78307 | Val Loss = 0.45164, Val Acc = 0.78614
Epoch 19: Train Loss = 0.45770, Train Acc = 0.78343 | Val Loss = 0.45209, Val Acc = 0.78581
Epoch 20: Train Loss = 0.45872, Train Acc = 0.78346 | Val Loss = 0.45278, Val Acc = 0.78444
Epoch 21: Train Loss = 0.45720, Train Acc = 0.78429 | Val Loss = 0.45394, Val Acc = 0.78548
Epoch 22: Train Loss = 0.45647, Train Acc = 0.78514 | Val Loss = 0.45336, Val Acc = 0.78522
Epoch 23: Train Loss = 0.45543, Train Acc = 0.78455 | Val Loss = 0.45191, Val Acc = 0.78573
Epoch 24: Train Loss = 0.45507, Train Acc = 0.78556 | Val Loss = 0.45369, Val Acc = 0.78389
Epoch 25: Train Loss = 0.45420, Train Acc = 0.78460 | Val Loss = 0.45290, Val Acc = 0.78571
Epoch 26: Train Loss = 0.45436, Train Acc = 0.78630 | Val Loss = 0.45362, Val Acc = 0.78441
Epoch 27: Train Loss = 0.45400, Train Acc = 0.78556 | Val Loss = 0.45346, Val Acc = 0.78427
Epoch 28: Train Loss = 0.45284, Train Acc = 0.78641 | Val Loss = 0.45251, Val Acc = 0.78533
Epoch 29: Train Loss = 0.45318, Train Acc = 0.78650 | Val Loss = 0.45107, Val Acc = 0.78644
Epoch 30: Train Loss = 0.45119, Train Acc = 0.78717 | Val Loss = 0.45163, Val Acc = 0.78545
```




```
Accuracy: 0.785451
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.78      0.78     21197
           1       0.78      0.79      0.79     21199

    accuracy                           0.79     42396
   macro avg       0.79      0.79      0.79     42396
weighted avg       0.79      0.79      0.79     42396
```

Figure 11: Experiment 11

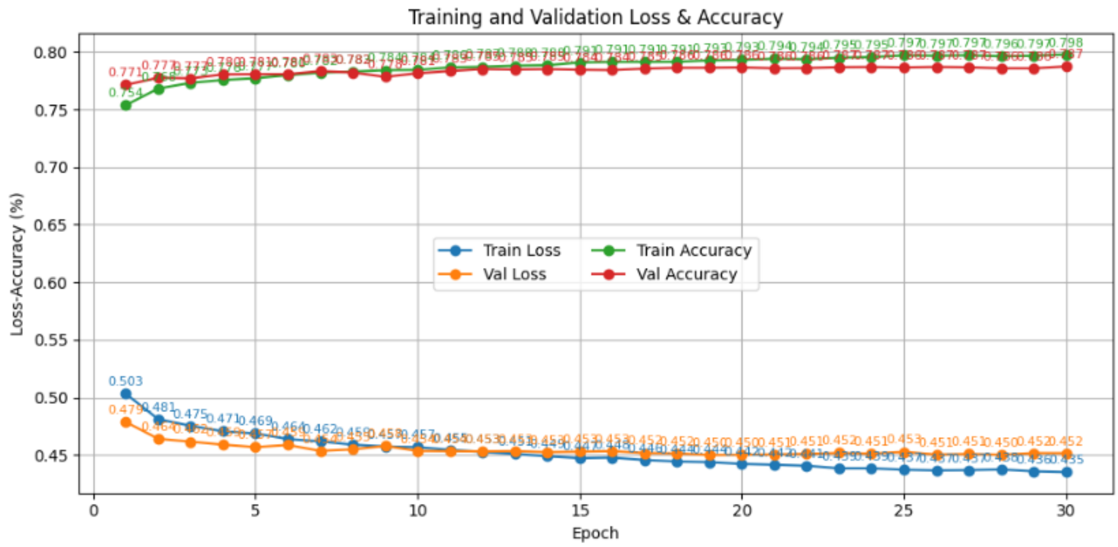
- The results indicate similar metrics for the accuracy and classification report, as a larger batch size means faster training **Experiment 9** is still considered the best.

12. Twelfth Experiment - Increase Batch Size

    - This experiment uses the same configuration as **Experiment 9** but with `batch_size=256` instead of `batch_size=128`.

- Results:

```
Epoch  1: Train Loss = 0.50316, Train Acc = 0.75371 | Val Loss = 0.47872, Val Acc = 0.77125
Epoch  2: Train Loss = 0.48117, Train Acc = 0.76809 | Val Loss = 0.46417, Val Acc = 0.77746
Epoch  3: Train Loss = 0.47545, Train Acc = 0.77306 | Val Loss = 0.46167, Val Acc = 0.77691
Epoch  4: Train Loss = 0.47065, Train Acc = 0.77551 | Val Loss = 0.45907, Val Acc = 0.78036
Epoch  5: Train Loss = 0.46879, Train Acc = 0.77702 | Val Loss = 0.45695, Val Acc = 0.78055
Epoch  6: Train Loss = 0.46396, Train Acc = 0.77967 | Val Loss = 0.45879, Val Acc = 0.78052
Epoch  7: Train Loss = 0.46188, Train Acc = 0.78191 | Val Loss = 0.45367, Val Acc = 0.78319
Epoch  8: Train Loss = 0.45907, Train Acc = 0.78270 | Val Loss = 0.45501, Val Acc = 0.78198
Epoch  9: Train Loss = 0.45750, Train Acc = 0.78402 | Val Loss = 0.45794, Val Acc = 0.77823
Epoch 10: Train Loss = 0.45688, Train Acc = 0.78425 | Val Loss = 0.45368, Val Acc = 0.78135
Epoch 11: Train Loss = 0.45458, Train Acc = 0.78649 | Val Loss = 0.45351, Val Acc = 0.78326
Epoch 12: Train Loss = 0.45283, Train Acc = 0.78695 | Val Loss = 0.45327, Val Acc = 0.78500
Epoch 13: Train Loss = 0.45107, Train Acc = 0.78793 | Val Loss = 0.45349, Val Acc = 0.78456
Epoch 14: Train Loss = 0.44904, Train Acc = 0.78834 | Val Loss = 0.45262, Val Acc = 0.78496
Epoch 15: Train Loss = 0.44739, Train Acc = 0.79072 | Val Loss = 0.45303, Val Acc = 0.78437
Epoch 16: Train Loss = 0.44788, Train Acc = 0.79117 | Val Loss = 0.45349, Val Acc = 0.78408
Epoch 17: Train Loss = 0.44578, Train Acc = 0.79117 | Val Loss = 0.45186, Val Acc = 0.78538
Epoch 18: Train Loss = 0.44449, Train Acc = 0.79126 | Val Loss = 0.45155, Val Acc = 0.78609
Epoch 19: Train Loss = 0.44392, Train Acc = 0.79253 | Val Loss = 0.45009, Val Acc = 0.78616
Epoch 20: Train Loss = 0.44239, Train Acc = 0.79302 | Val Loss = 0.45025, Val Acc = 0.78644
Epoch 21: Train Loss = 0.44167, Train Acc = 0.79395 | Val Loss = 0.45063, Val Acc = 0.78571
Epoch 22: Train Loss = 0.44078, Train Acc = 0.79357 | Val Loss = 0.45108, Val Acc = 0.78590
Epoch 23: Train Loss = 0.43856, Train Acc = 0.79484 | Val Loss = 0.45189, Val Acc = 0.78663
Epoch 24: Train Loss = 0.43856, Train Acc = 0.79534 | Val Loss = 0.45133, Val Acc = 0.78677
Epoch 25: Train Loss = 0.43733, Train Acc = 0.79686 | Val Loss = 0.45282, Val Acc = 0.78639
Epoch 26: Train Loss = 0.43675, Train Acc = 0.79658 | Val Loss = 0.45053, Val Acc = 0.78684
Epoch 27: Train Loss = 0.43701, Train Acc = 0.79694 | Val Loss = 0.45101, Val Acc = 0.78654
Epoch 28: Train Loss = 0.43758, Train Acc = 0.79610 | Val Loss = 0.45030, Val Acc = 0.78569
Epoch 29: Train Loss = 0.43598, Train Acc = 0.79652 | Val Loss = 0.45178, Val Acc = 0.78566
Epoch 30: Train Loss = 0.43527, Train Acc = 0.79761 | Val Loss = 0.45172, Val Acc = 0.78727
```




```
Accuracy: 0.787268
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.81      0.79     21197
           1       0.80      0.77      0.78     21199

    accuracy                           0.79     42396
   macro avg       0.79      0.79      0.79     42396
weighted avg       0.79      0.79      0.79     42396
```

Figure 12: Experiment 12

- The results indicate similar accuracy metrics, but the classification report is less stable. **Experiment 9** is still considered the best.

13. Thirteen Experiment - Optimization

- To optimize the hyper-parameters of **Experiment 9** the framework **Optuna** was used for 50 runs of 20 epochs.

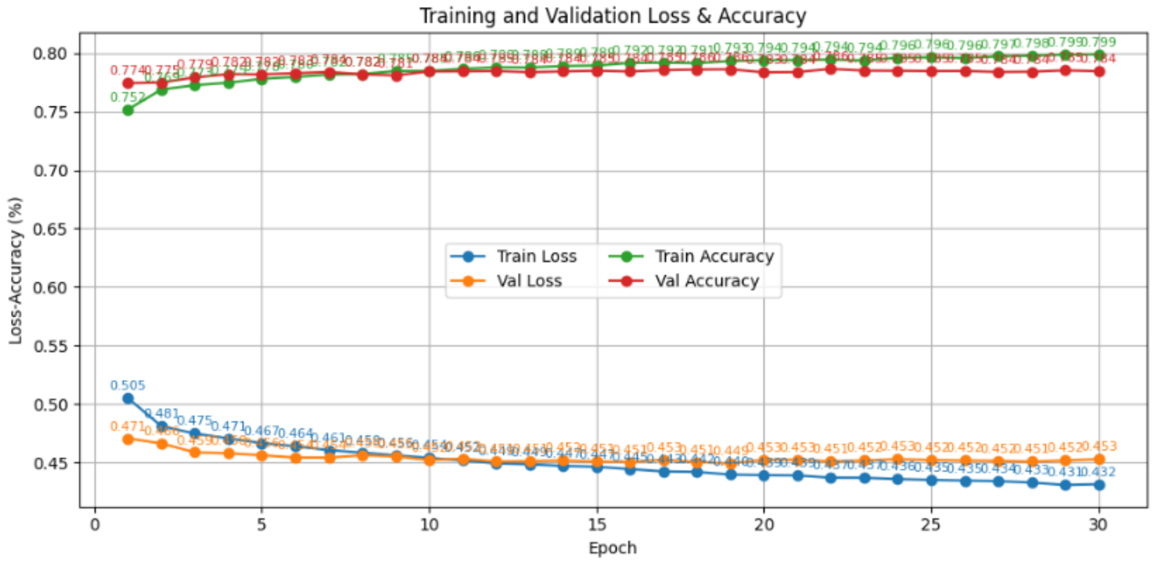
- The following are the parameters tested by Optuna:
  (a) `optimizer_name` -> ["Adam", "AdamW"]
  (b) `learning_rate` -> floats between 1e-3 and 1e-1
  (c) `weight_decay` -> floats between 1e-6 and 1e-4
  (d) `dropout_prob` -> floats between 0.25 and 0.5
- The value ranges were selected using a combination of previous draft runs of Optuna and manual testing.
- Optimization results:

```
Epoch  19: Train Loss = 0.44887, Train Acc = 0.78990 | Val Loss = 0.45679, Val Acc = 0.78132
[I 2025-04-15 11:12:04,795] Trial 48 finished with value: 0.45539774117340526 and parameters: {'optimizer_name': 'AdamW', 'learning_rat
e': 0.0125469966707564, 'weight_decay': 1.8716483219033318e-05, 'dropout_prob': 0.35991237149788785}. Best is trial 34 with value: 0.45
30378578298063.
Epoch  20: Train Loss = 0.44942, Train Acc = 0.78840 | Val Loss = 0.45540, Val Acc = 0.78257
Epoch   1: Train Loss = 0.50117, Train Acc = 0.75442 | Val Loss = 0.47672, Val Acc = 0.76757
Epoch   2: Train Loss = 0.48346, Train Acc = 0.76712 | Val Loss = 0.46545, Val Acc = 0.77500
Epoch   3: Train Loss = 0.47710, Train Acc = 0.77141 | Val Loss = 0.46526, Val Acc = 0.77753
Epoch   4: Train Loss = 0.47280, Train Acc = 0.77348 | Val Loss = 0.45901, Val Acc = 0.77941
Epoch   5: Train Loss = 0.46847, Train Acc = 0.77731 | Val Loss = 0.45891, Val Acc = 0.78047
Epoch   6: Train Loss = 0.46529, Train Acc = 0.77826 | Val Loss = 0.46217, Val Acc = 0.77852
Epoch   7: Train Loss = 0.46167, Train Acc = 0.78050 | Val Loss = 0.45864, Val Acc = 0.77970
Epoch   8: Train Loss = 0.45898, Train Acc = 0.78145 | Val Loss = 0.45780, Val Acc = 0.77939
Epoch   9: Train Loss = 0.45671, Train Acc = 0.78247 | Val Loss = 0.45710, Val Acc = 0.78111
Epoch  10: Train Loss = 0.45490, Train Acc = 0.78336 | Val Loss = 0.45596, Val Acc = 0.78201
Epoch  11: Train Loss = 0.45275, Train Acc = 0.78507 | Val Loss = 0.45611, Val Acc = 0.78250
Epoch  12: Train Loss = 0.45071, Train Acc = 0.78567 | Val Loss = 0.45508, Val Acc = 0.78314
Epoch  13: Train Loss = 0.44888, Train Acc = 0.78787 | Val Loss = 0.45640, Val Acc = 0.78342
Epoch  14: Train Loss = 0.44845, Train Acc = 0.78762 | Val Loss = 0.45387, Val Acc = 0.78529
Epoch  15: Train Loss = 0.44450, Train Acc = 0.78952 | Val Loss = 0.45591, Val Acc = 0.78205
Epoch  16: Train Loss = 0.44440, Train Acc = 0.78973 | Val Loss = 0.45536, Val Acc = 0.78356
Epoch  17: Train Loss = 0.44251, Train Acc = 0.79236 | Val Loss = 0.45422, Val Acc = 0.78411
Epoch  18: Train Loss = 0.44156, Train Acc = 0.79149 | Val Loss = 0.45592, Val Acc = 0.78385
Epoch  19: Train Loss = 0.44054, Train Acc = 0.79226 | Val Loss = 0.45428, Val Acc = 0.78290
[I 2025-04-15 11:13:36,214] Trial 49 finished with value: 0.4588513873427747 and parameters: {'optimizer_name': 'AdamW', 'learning_rat
e': 0.00286422029302957, 'weight_decay': 2.7453163998042397e-05, 'dropout_prob': 0.3296590440397862}. Best is trial 34 with value: 0.4
530378578298063.
Epoch  20: Train Loss = 0.43902, Train Acc = 0.79371 | Val Loss = 0.45885, Val Acc = 0.78130
numbers of the finished trials: 50
the best params: {'optimizer_name': 'AdamW', 'learning_rate': 0.0010291927407311168, 'weight_decay': 8.018811458914011e-06, 'dropout_pr
ob': 0.3553718703420643}
the best value: 0.4530378578298063
Optimization Process - Finished
```

Figure 13: Optimization Results

- Results:

```
Epoch  1: Train Loss = 0.50514, Train Acc = 0.75172 | Val Loss = 0.47075, Val Acc = 0.77422
Epoch  2: Train Loss = 0.48119, Train Acc = 0.76885 | Val Loss = 0.46649, Val Acc = 0.77486
Epoch  3: Train Loss = 0.47491, Train Acc = 0.77250 | Val Loss = 0.45887, Val Acc = 0.77911
Epoch  4: Train Loss = 0.47083, Train Acc = 0.77460 | Val Loss = 0.45806, Val Acc = 0.78196
Epoch  5: Train Loss = 0.46676, Train Acc = 0.77798 | Val Loss = 0.45636, Val Acc = 0.78170
Epoch  6: Train Loss = 0.46409, Train Acc = 0.77953 | Val Loss = 0.45445, Val Acc = 0.78274
Epoch  7: Train Loss = 0.46076, Train Acc = 0.78159 | Val Loss = 0.45440, Val Acc = 0.78378
Epoch  8: Train Loss = 0.45861, Train Acc = 0.78206 | Val Loss = 0.45633, Val Acc = 0.78180
Epoch  9: Train Loss = 0.45632, Train Acc = 0.78456 | Val Loss = 0.45541, Val Acc = 0.78088
Epoch 10: Train Loss = 0.45412, Train Acc = 0.78467 | Val Loss = 0.45227, Val Acc = 0.78425
Epoch 11: Train Loss = 0.45192, Train Acc = 0.78646 | Val Loss = 0.45348, Val Acc = 0.78434
Epoch 12: Train Loss = 0.44946, Train Acc = 0.78779 | Val Loss = 0.45106, Val Acc = 0.78460
Epoch 13: Train Loss = 0.44870, Train Acc = 0.78762 | Val Loss = 0.45133, Val Acc = 0.78364
Epoch 14: Train Loss = 0.44729, Train Acc = 0.78888 | Val Loss = 0.45193, Val Acc = 0.78427
Epoch 15: Train Loss = 0.44656, Train Acc = 0.78931 | Val Loss = 0.45103, Val Acc = 0.78486
Epoch 16: Train Loss = 0.44457, Train Acc = 0.79157 | Val Loss = 0.45073, Val Acc = 0.78420
Epoch 17: Train Loss = 0.44257, Train Acc = 0.79174 | Val Loss = 0.45251, Val Acc = 0.78538
Epoch 18: Train Loss = 0.44218, Train Acc = 0.79134 | Val Loss = 0.45065, Val Acc = 0.78588
Epoch 19: Train Loss = 0.43986, Train Acc = 0.79312 | Val Loss = 0.44941, Val Acc = 0.78609
Epoch 20: Train Loss = 0.43940, Train Acc = 0.79393 | Val Loss = 0.45258, Val Acc = 0.78349
Epoch 21: Train Loss = 0.43914, Train Acc = 0.79397 | Val Loss = 0.45259, Val Acc = 0.78373
Epoch 22: Train Loss = 0.43723, Train Acc = 0.79440 | Val Loss = 0.45117, Val Acc = 0.78642
Epoch 23: Train Loss = 0.43714, Train Acc = 0.79368 | Val Loss = 0.45187, Val Acc = 0.78496
Epoch 24: Train Loss = 0.43605, Train Acc = 0.79595 | Val Loss = 0.45309, Val Acc = 0.78484
Epoch 25: Train Loss = 0.43537, Train Acc = 0.79624 | Val Loss = 0.45229, Val Acc = 0.78458
Epoch 26: Train Loss = 0.43477, Train Acc = 0.79577 | Val Loss = 0.45192, Val Acc = 0.78465
Epoch 27: Train Loss = 0.43428, Train Acc = 0.79710 | Val Loss = 0.45163, Val Acc = 0.78382
Epoch 28: Train Loss = 0.43306, Train Acc = 0.79766 | Val Loss = 0.45093, Val Acc = 0.78404
Epoch 29: Train Loss = 0.43101, Train Acc = 0.79861 | Val Loss = 0.45199, Val Acc = 0.78517
Epoch 30: Train Loss = 0.43168, Train Acc = 0.79889 | Val Loss = 0.45302, Val Acc = 0.78446
```




```
Accuracy: 0.784461
Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.76      0.78     21197
           1       0.77      0.81      0.79     21199

    accuracy                           0.78     42396
   macro avg       0.79      0.78      0.78     42396
weighted avg       0.79      0.78      0.78     42396
```

Figure 14: Experiment 13

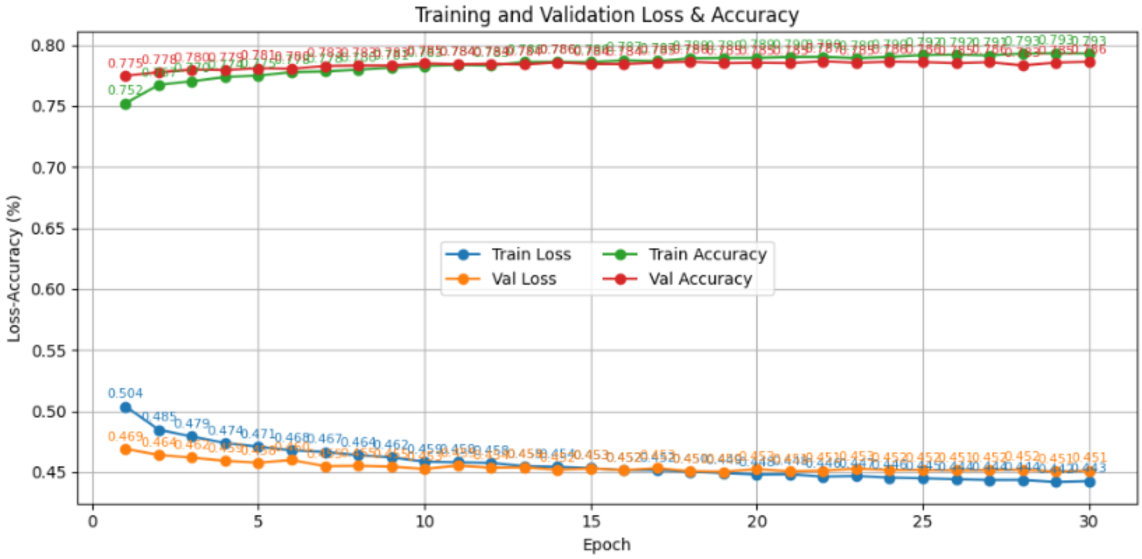
- The results show a clear imbalance in the classification report metrics and overall lower values, for this reason, the best model of the experiments will be considered the one of **Experiment 9**.

3.2. Hyper-parameter tuning

1. Final Results:

    - The best results are the ones of `Experiment 9`.

```
Train and Evaluate Best Model - Started
Epoch  1: Train Loss = 0.50387, Train Acc = 0.75216 | Val Loss = 0.46909, Val Acc = 0.77479
Epoch  2: Train Loss = 0.48481, Train Acc = 0.76723 | Val Loss = 0.46415, Val Acc = 0.77755
Epoch  3: Train Loss = 0.47918, Train Acc = 0.77018 | Val Loss = 0.46190, Val Acc = 0.77972
Epoch  4: Train Loss = 0.47392, Train Acc = 0.77365 | Val Loss = 0.45934, Val Acc = 0.77934
Epoch  5: Train Loss = 0.47088, Train Acc = 0.77485 | Val Loss = 0.45783, Val Acc = 0.78085
Epoch  6: Train Loss = 0.46796, Train Acc = 0.77766 | Val Loss = 0.46000, Val Acc = 0.78038
Epoch  7: Train Loss = 0.46659, Train Acc = 0.77823 | Val Loss = 0.45498, Val Acc = 0.78272
Epoch  8: Train Loss = 0.46400, Train Acc = 0.77963 | Val Loss = 0.45526, Val Acc = 0.78305
Epoch  9: Train Loss = 0.46213, Train Acc = 0.78122 | Val Loss = 0.45462, Val Acc = 0.78305
Epoch 10: Train Loss = 0.45867, Train Acc = 0.78277 | Val Loss = 0.45270, Val Acc = 0.78481
Epoch 11: Train Loss = 0.45814, Train Acc = 0.78363 | Val Loss = 0.45548, Val Acc = 0.78401
Epoch 12: Train Loss = 0.45763, Train Acc = 0.78310 | Val Loss = 0.45371, Val Acc = 0.78448
Epoch 13: Train Loss = 0.45494, Train Acc = 0.78592 | Val Loss = 0.45403, Val Acc = 0.78394
Epoch 14: Train Loss = 0.45440, Train Acc = 0.78607 | Val Loss = 0.45203, Val Acc = 0.78562
Epoch 15: Train Loss = 0.45331, Train Acc = 0.78584 | Val Loss = 0.45319, Val Acc = 0.78441
Epoch 16: Train Loss = 0.45164, Train Acc = 0.78737 | Val Loss = 0.45172, Val Acc = 0.78432
Epoch 17: Train Loss = 0.45174, Train Acc = 0.78664 | Val Loss = 0.45333, Val Acc = 0.78540
Epoch 18: Train Loss = 0.45030, Train Acc = 0.78906 | Val Loss = 0.45080, Val Acc = 0.78614
Epoch 19: Train Loss = 0.44937, Train Acc = 0.78926 | Val Loss = 0.45039, Val Acc = 0.78496
Epoch 20: Train Loss = 0.44817, Train Acc = 0.78932 | Val Loss = 0.45260, Val Acc = 0.78533
Epoch 21: Train Loss = 0.44834, Train Acc = 0.79005 | Val Loss = 0.45071, Val Acc = 0.78500
Epoch 22: Train Loss = 0.44642, Train Acc = 0.79017 | Val Loss = 0.45133, Val Acc = 0.78654
Epoch 23: Train Loss = 0.44692, Train Acc = 0.78913 | Val Loss = 0.45300, Val Acc = 0.78529
Epoch 24: Train Loss = 0.44560, Train Acc = 0.79009 | Val Loss = 0.45187, Val Acc = 0.78618
Epoch 25: Train Loss = 0.44502, Train Acc = 0.79159 | Val Loss = 0.45190, Val Acc = 0.78588
Epoch 26: Train Loss = 0.44423, Train Acc = 0.79196 | Val Loss = 0.45149, Val Acc = 0.78491
Epoch 27: Train Loss = 0.44366, Train Acc = 0.79129 | Val Loss = 0.45177, Val Acc = 0.78576
Epoch 28: Train Loss = 0.44371, Train Acc = 0.79292 | Val Loss = 0.45212, Val Acc = 0.78319
Epoch 29: Train Loss = 0.44198, Train Acc = 0.79312 | Val Loss = 0.45056, Val Acc = 0.78540
Epoch 30: Train Loss = 0.44275, Train Acc = 0.79297 | Val Loss = 0.45130, Val Acc = 0.78616
```




```
Accuracy: 0.786159
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.78      0.78     21197
           1       0.78      0.80      0.79     21199

    accuracy                           0.79     42396
   macro avg       0.79      0.79      0.79     42396
weighted avg       0.79      0.79      0.79     42396
```

Figure 15: Best Model - Results

- The results seem to get around the same value of 0.79 for all metrics, this demonstrates a balanced performance and indicates an effective classification without significant bias. The insignificant amount of overfitting can also be observed by the small `Delta` between the `Train` and `Val` curves of the plot above.

2. Model Configuration:

   - The final configuration is the following:
     - `batch_size` -> 128
     - `network architecture` -> Two hidden layers with ReLu as an activation function, batch optimization, and dropout.
     - `num_epochs` -> 30
     - `loss_func` -> nn.CrossEntropyLoss()
     - `optimizer_name` -> Adam
     - `learning_rate` -> 0.0019524310388177159
     - `weight_decay` -> 5.263544573640884e-06
     - `dropout_prob` -> 0.4125125247127012

3. Under-Fitting & Over-Fitting:

   - No under-fitting was observed in all of the experiments.
   - The `Delta` between the train and val accuracies is small enough to not be considered as overfitting.

3.3. Optimization techniques

1. Optuna, an optimization framework, was used to run multiple times with different hyperparameters for a number of epochs to find the best match (lowest val loss), although not far from the best experiment, the results of Optuna were rejected for the reasons described above. More info about the use of Optuna on **Experiment 13**.

2. Batch normalization was done by normalizing the outputs of each layer to keep the values stable during training, which makes learning faster and more reliable.

3. Dropout was done by randomly turning off a percentage of neurons during training, helping to reduce overfitting by preventing the network from relying on specific neurons too much.

3.4. Evaluation

- I evaluated the predictions using accuracy, precision, recall, F1-score, and Plots.

- All the metrics reached around 78%, showing the general stability and good performance of the model. These results, and especially the F1-score, which can be described as the harmonic mean of the precision and recall of a classification model, reveal a good performance in recognizing positive cases while minimizing false positives and false negatives.
- Learning curves were plotted to analyze model performance across different experiments, helping to detect overfitting or underfitting trends.
- The following table shows the average metrics between the two classes of each experiment.

Experiment	Accuracy	Precision	Recall	F1-Score
1	0.71	0.71	0.71	0.71
2	0.76	0.76	0.76	0.76
3	0.78	0.78	0.78	0.78
4	0.79	0.79	0.79	0.79
5	0.78	0.78	0.78	0.78
6	0.78	0.78	0.78	0.78
7	0.78	0.78	0.78	0.78
8	0.79	0.79	0.79	0.79
9	0.79	0.79	0.79	0.79
10	0.79	0.79	0.79	0.79
11	0.79	0.79	0.79	0.79
12	0.79	0.79	0.79	0.79
13	0.79	0.79	0.79	0.79

Table 1: Experiments

3.4.1. ROC curve.

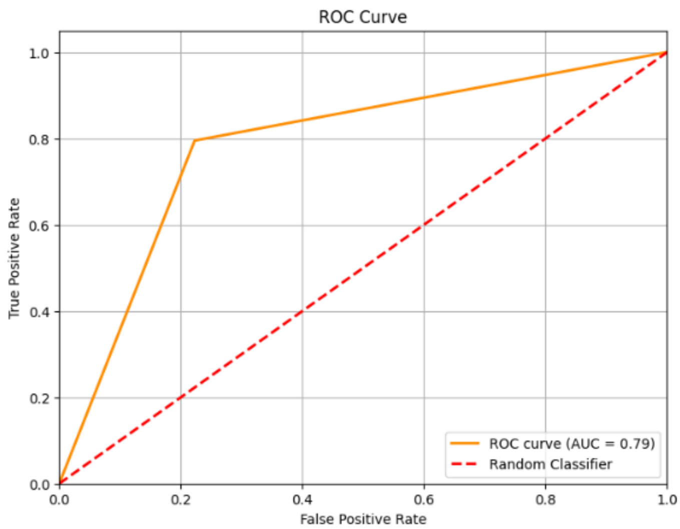


Figure 16: Best Model - ROC Curve

- The ROC (Receiver Operating Characteristic) curve shows that the classifier achieves a TPR (True Positive Rate) of about 0.80 at a FPR (False Positive Rate) of around 0.20-0.25, rising well above the "random" baseline classifier.
- An overall AUC (Area Under the ROC Curve) of 0.79 indicates that the model has good discrimination ability between positive and negative tweets, correctly ranking a randomly chosen positive example above a negative one roughly 79% of the time.

3.4.2. Learning Curve.

```
Train and Evaluate Best Model - Started
Epoch  1: Train Loss = 0.50387, Train Acc = 0.75216 | Val Loss = 0.46909, Val Acc = 0.77479
Epoch  2: Train Loss = 0.48481, Train Acc = 0.76723 | Val Loss = 0.46415, Val Acc = 0.77755
Epoch  3: Train Loss = 0.47918, Train Acc = 0.77018 | Val Loss = 0.46190, Val Acc = 0.77972
Epoch  4: Train Loss = 0.47392, Train Acc = 0.77365 | Val Loss = 0.45934, Val Acc = 0.77934
Epoch  5: Train Loss = 0.47088, Train Acc = 0.77485 | Val Loss = 0.45783, Val Acc = 0.78085
Epoch  6: Train Loss = 0.46796, Train Acc = 0.77766 | Val Loss = 0.46000, Val Acc = 0.78038
Epoch  7: Train Loss = 0.46659, Train Acc = 0.77823 | Val Loss = 0.45498, Val Acc = 0.78272
Epoch  8: Train Loss = 0.46400, Train Acc = 0.77963 | Val Loss = 0.45526, Val Acc = 0.78305
Epoch  9: Train Loss = 0.46213, Train Acc = 0.78122 | Val Loss = 0.45462, Val Acc = 0.78305
Epoch 10: Train Loss = 0.45867, Train Acc = 0.78277 | Val Loss = 0.45270, Val Acc = 0.78481
Epoch 11: Train Loss = 0.45814, Train Acc = 0.78363 | Val Loss = 0.45548, Val Acc = 0.78401
Epoch 12: Train Loss = 0.45763, Train Acc = 0.78310 | Val Loss = 0.45371, Val Acc = 0.78448
Epoch 13: Train Loss = 0.45494, Train Acc = 0.78592 | Val Loss = 0.45403, Val Acc = 0.78394
Epoch 14: Train Loss = 0.45440, Train Acc = 0.78607 | Val Loss = 0.45203, Val Acc = 0.78562
Epoch 15: Train Loss = 0.45331, Train Acc = 0.78584 | Val Loss = 0.45319, Val Acc = 0.78441
Epoch 16: Train Loss = 0.45164, Train Acc = 0.78737 | Val Loss = 0.45172, Val Acc = 0.78432
Epoch 17: Train Loss = 0.45174, Train Acc = 0.78664 | Val Loss = 0.45333, Val Acc = 0.78540
Epoch 18: Train Loss = 0.45030, Train Acc = 0.78906 | Val Loss = 0.45080, Val Acc = 0.78614
Epoch 19: Train Loss = 0.44937, Train Acc = 0.78926 | Val Loss = 0.45039, Val Acc = 0.78496
Epoch 20: Train Loss = 0.44817, Train Acc = 0.78932 | Val Loss = 0.45260, Val Acc = 0.78533
Epoch 21: Train Loss = 0.44834, Train Acc = 0.79005 | Val Loss = 0.45071, Val Acc = 0.78500
Epoch 22: Train Loss = 0.44642, Train Acc = 0.79017 | Val Loss = 0.45133, Val Acc = 0.78654
Epoch 23: Train Loss = 0.44692, Train Acc = 0.78913 | Val Loss = 0.45300, Val Acc = 0.78529
Epoch 24: Train Loss = 0.44560, Train Acc = 0.79009 | Val Loss = 0.45187, Val Acc = 0.78618
Epoch 25: Train Loss = 0.44502, Train Acc = 0.79159 | Val Loss = 0.45190, Val Acc = 0.78588
Epoch 26: Train Loss = 0.44423, Train Acc = 0.79196 | Val Loss = 0.45149, Val Acc = 0.78491
Epoch 27: Train Loss = 0.44366, Train Acc = 0.79129 | Val Loss = 0.45177, Val Acc = 0.78576
Epoch 28: Train Loss = 0.44371, Train Acc = 0.79292 | Val Loss = 0.45212, Val Acc = 0.78319
Epoch 29: Train Loss = 0.44198, Train Acc = 0.79312 | Val Loss = 0.45056, Val Acc = 0.78540
Epoch 30: Train Loss = 0.44275, Train Acc = 0.79297 | Val Loss = 0.45130, Val Acc = 0.78616
```

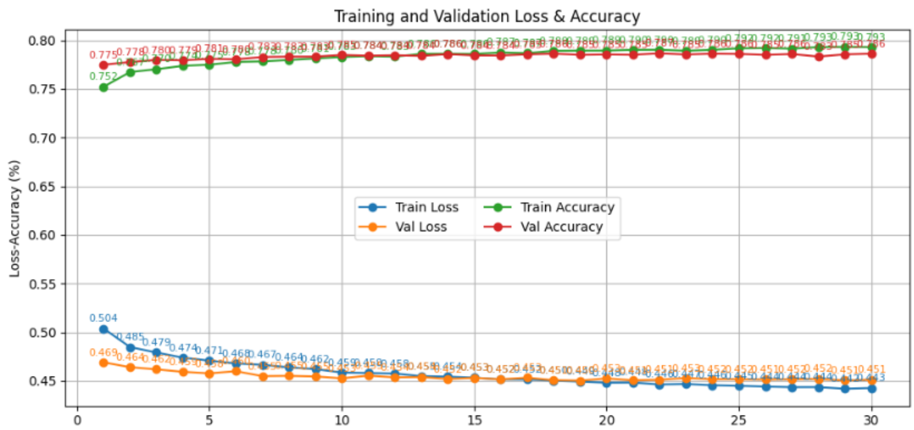


Figure 17: Best Model - Learning Curve

- The **training curves** show a steady drop in **loss** from about **0.50** to **0.44** over **30** epochs, with **training accuracy** climbing from around **75%** up to **79%**.
- The **validation curves** mirror this trend, **val loss** falls from around **0.47** to **0.45** and **val accuracy** rises from around **77%** to **79%**, both show most improvement at roughly the first **10 epochs**, improvement is still present afterwards but with a slower rate.
- The small but persistent gap (**training acc** around **79%** vs. **validation acc** around **78.5%**) indicates only mild overfitting, and the fact that both **losses** continue to

decrease alongside each other suggests the model is well-tuned and not underfitting.

3.4.3. Confusion matrix.

- True Negatives (TN): 16.474
- False Positives (FP): 4.723
- False Negatives (FN): 4.343
- True Positives (TP): 16.856

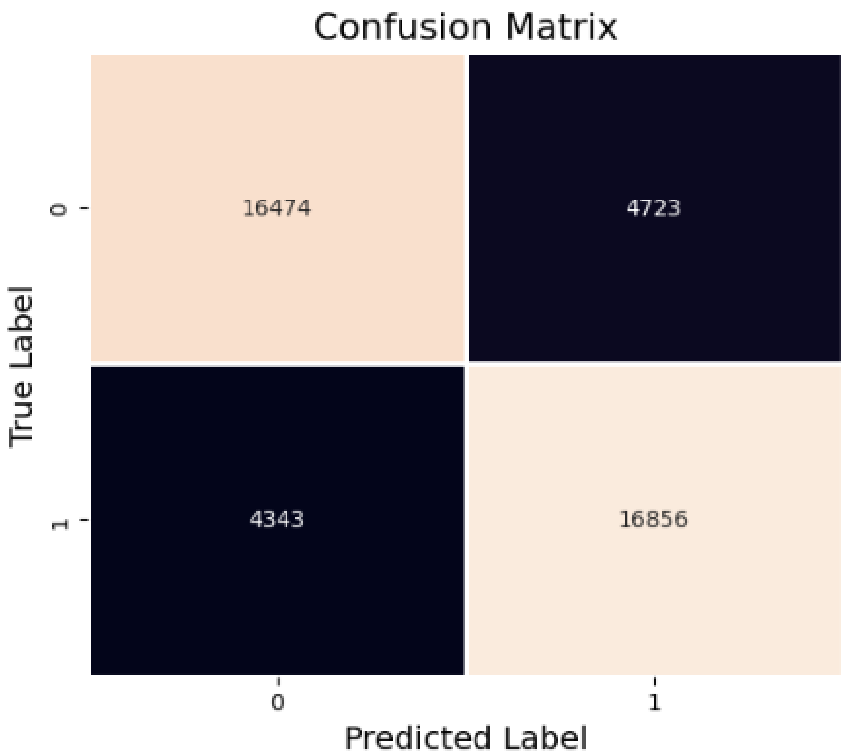


Figure 18: Best Model - Confusion Matrix

- The model performs well overall, as the numbers for correct predictions (TP & TN) are significantly higher than incorrect ones.

4. Results and Overall Analysis

4.1. Results Analysis

- My final results show an accuracy of around 78.5%, meaning that the model makes a correct sentiment prediction 78.5% of the time. Metrics such as precision, recall, and f1-score, alongside the Learning Curves plot, also show that the model is balanced.

- More experiments I would make would be to run the Optuna optimization for a number of times and epochs for each one of the experiments made.

4.1.1. Best trial.

- Results of best trial:

```
Accuracy: 0.786159
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.78      0.78     21197
           1       0.78      0.80      0.79     21199

    accuracy                           0.79     42396
   macro avg       0.79      0.79      0.79     42396
weighted avg       0.79      0.79      0.79     42396
```

Figure 19: Best Trial - Classification Report

```
Train and Evaluate Best Model - Started
Epoch   1: Train Loss = 0.50387, Train Acc = 0.75216 | Val Loss = 0.46909, Val Acc = 0.77479
Epoch   2: Train Loss = 0.48481, Train Acc = 0.76723 | Val Loss = 0.46415, Val Acc = 0.77755
Epoch   3: Train Loss = 0.47918, Train Acc = 0.77018 | Val Loss = 0.46190, Val Acc = 0.77972
Epoch   4: Train Loss = 0.47392, Train Acc = 0.77365 | Val Loss = 0.45934, Val Acc = 0.77934
Epoch   5: Train Loss = 0.47088, Train Acc = 0.77485 | Val Loss = 0.45783, Val Acc = 0.78085
Epoch   6: Train Loss = 0.46796, Train Acc = 0.77766 | Val Loss = 0.46000, Val Acc = 0.78038
Epoch   7: Train Loss = 0.46659, Train Acc = 0.77823 | Val Loss = 0.45498, Val Acc = 0.78272
Epoch   8: Train Loss = 0.46400, Train Acc = 0.77963 | Val Loss = 0.45526, Val Acc = 0.78305
Epoch   9: Train Loss = 0.46213, Train Acc = 0.78122 | Val Loss = 0.45462, Val Acc = 0.78305
Epoch  10: Train Loss = 0.45867, Train Acc = 0.78277 | Val Loss = 0.45270, Val Acc = 0.78481
Epoch  11: Train Loss = 0.45814, Train Acc = 0.78363 | Val Loss = 0.45548, Val Acc = 0.78401
Epoch  12: Train Loss = 0.45763, Train Acc = 0.78310 | Val Loss = 0.45371, Val Acc = 0.78448
Epoch  13: Train Loss = 0.45494, Train Acc = 0.78592 | Val Loss = 0.45403, Val Acc = 0.78394
Epoch  14: Train Loss = 0.45440, Train Acc = 0.78607 | Val Loss = 0.45203, Val Acc = 0.78562
Epoch  15: Train Loss = 0.45331, Train Acc = 0.78584 | Val Loss = 0.45319, Val Acc = 0.78441
Epoch  16: Train Loss = 0.45164, Train Acc = 0.78737 | Val Loss = 0.45172, Val Acc = 0.78432
Epoch  17: Train Loss = 0.45174, Train Acc = 0.78664 | Val Loss = 0.45333, Val Acc = 0.78540
Epoch  18: Train Loss = 0.45030, Train Acc = 0.78906 | Val Loss = 0.45080, Val Acc = 0.78614
Epoch  19: Train Loss = 0.44937, Train Acc = 0.78926 | Val Loss = 0.45039, Val Acc = 0.78496
Epoch  20: Train Loss = 0.44817, Train Acc = 0.78932 | Val Loss = 0.45260, Val Acc = 0.78533
Epoch  21: Train Loss = 0.44834, Train Acc = 0.79005 | Val Loss = 0.45071, Val Acc = 0.78500
Epoch  22: Train Loss = 0.44642, Train Acc = 0.79017 | Val Loss = 0.45133, Val Acc = 0.78654
Epoch  23: Train Loss = 0.44692, Train Acc = 0.78913 | Val Loss = 0.45300, Val Acc = 0.78529
Epoch  24: Train Loss = 0.44560, Train Acc = 0.79009 | Val Loss = 0.45187, Val Acc = 0.78618
Epoch  25: Train Loss = 0.44502, Train Acc = 0.79159 | Val Loss = 0.45190, Val Acc = 0.78588
Epoch  26: Train Loss = 0.44423, Train Acc = 0.79196 | Val Loss = 0.45149, Val Acc = 0.78491
Epoch  27: Train Loss = 0.44366, Train Acc = 0.79129 | Val Loss = 0.45177, Val Acc = 0.78576
Epoch  28: Train Loss = 0.44371, Train Acc = 0.79292 | Val Loss = 0.45212, Val Acc = 0.78319
Epoch  29: Train Loss = 0.44198, Train Acc = 0.79312 | Val Loss = 0.45056, Val Acc = 0.78540
Epoch  30: Train Loss = 0.44275, Train Acc = 0.79297 | Val Loss = 0.45130, Val Acc = 0.78616
```

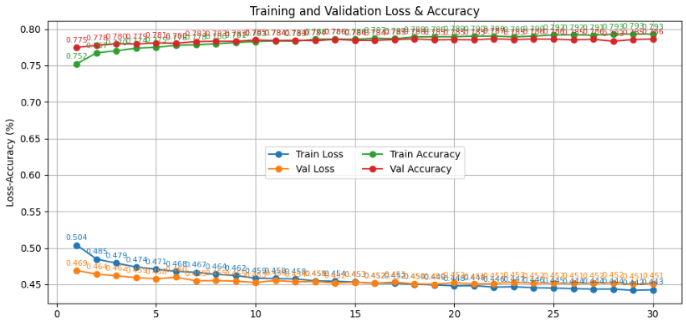


Figure 20: Best Trial - Learning Curve

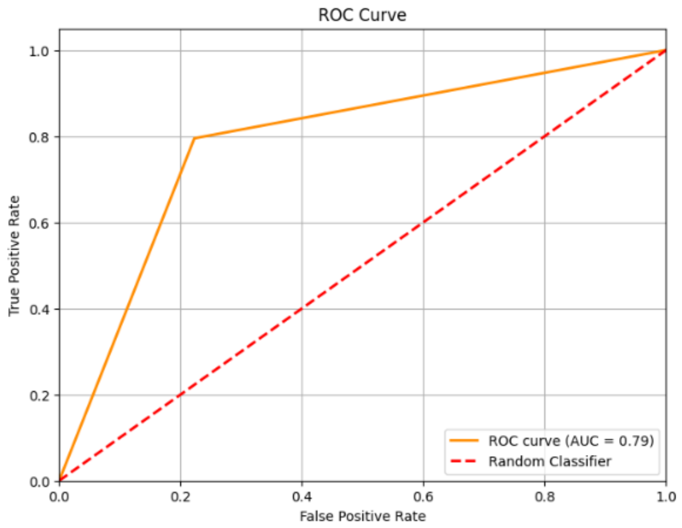


Figure 21: Best Trial - ROC Curve

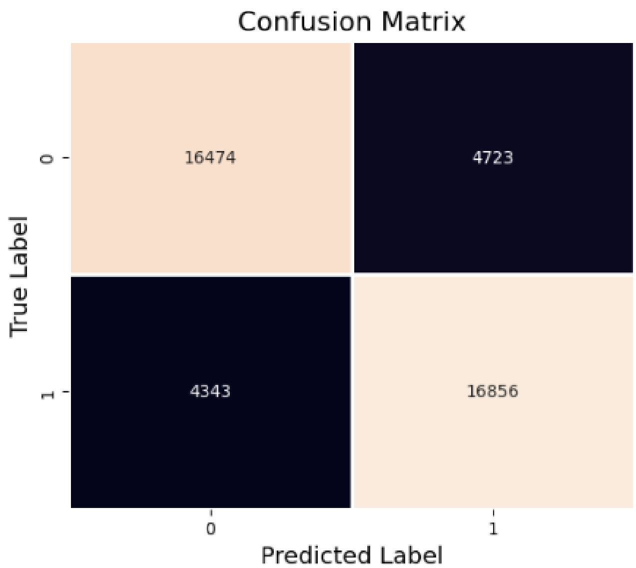


Figure 22: Best Trial - Confusion Matrix

- Look at **Experiment 9** and **3.2. Hyper-parameter tuning** for more information.

4.2. Comparison with the first project

- Overall, the neural-network model in Project 2 achieves a final accuracy of about 78.6% (with precision/recall/F1 all around 0.79), whereas the simpler logistic-regression baseline in Project 1 topped out closer to 80% validation accuracy (and the other metrics).
- That small gap in accuracy between then two procjects probably stems largely from the input vectors.

- More precisely, averaging Word2Vec embeddings smooths away the fine-grained n-gram signals that TF–IDF feeds into logistic regression.
- The two-layer network has far more parameters and a tougher, and more complex training landscape, so it usually needs either more data or more careful hyperparameter tuning to outperform a simple logistic regression, which has fewer parameters and a single, easy-to-find optimum.