ChatGPT Sentiment Analysis Report

Prepared By:

Name	ID
Rana Ihab	**20190207**
Mariam Amr	**20190520**
Norhan Abdelkader	**20190600**

Natural Language Processing Course

Semester 2, 2023

We tried two models which are LSTM and CNN. For each model we did the following:

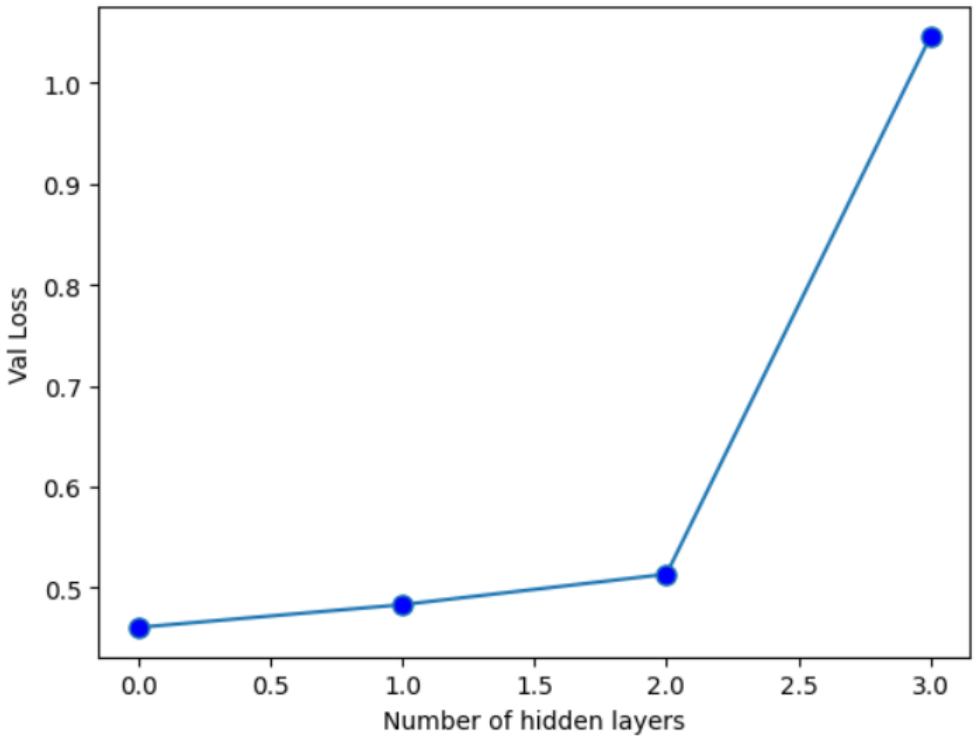
- Tuned each hyperparameter by trying different values for this parameter, while keeping the other parameters constant. (Evaluated by validation loss), we plot validation accuracy however, we tune by loss
- Try N architectures with a combination of the best hyperparameters. (Evaluated by test accuracy) to choose best overall model. Also, we tried 2 models with some of the bad hyperparameters to see their performance with different combination of parameters

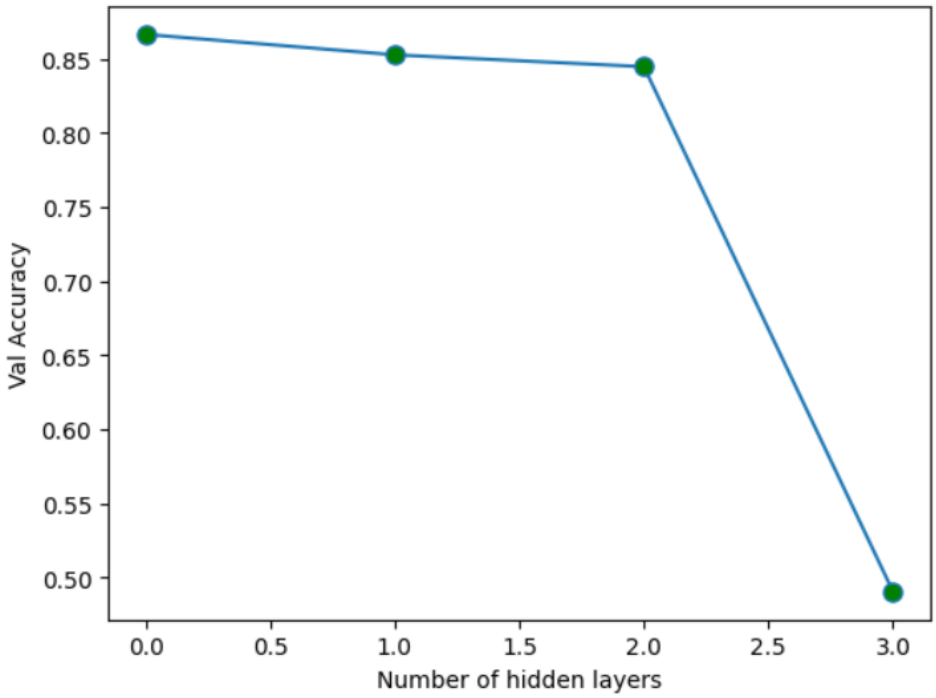
LSTM

1- Tune number of LSTM layers:
Values tried for hidden layers = [1, 2, 3, 4] (The graph plots number of hidden -1)

The other hyperparameters are constant and are:

1- Learning rate = 0.001
2- Batch size= 128
3- Number of units in lstm layer are [64, 32, 64, 32] where if it has only one layer then then it would be the last number.
4- Activation function in lstm layer is tanh
5- Regulizer L2 = 0.02
6- One dense layer of 3 units with softmax activation





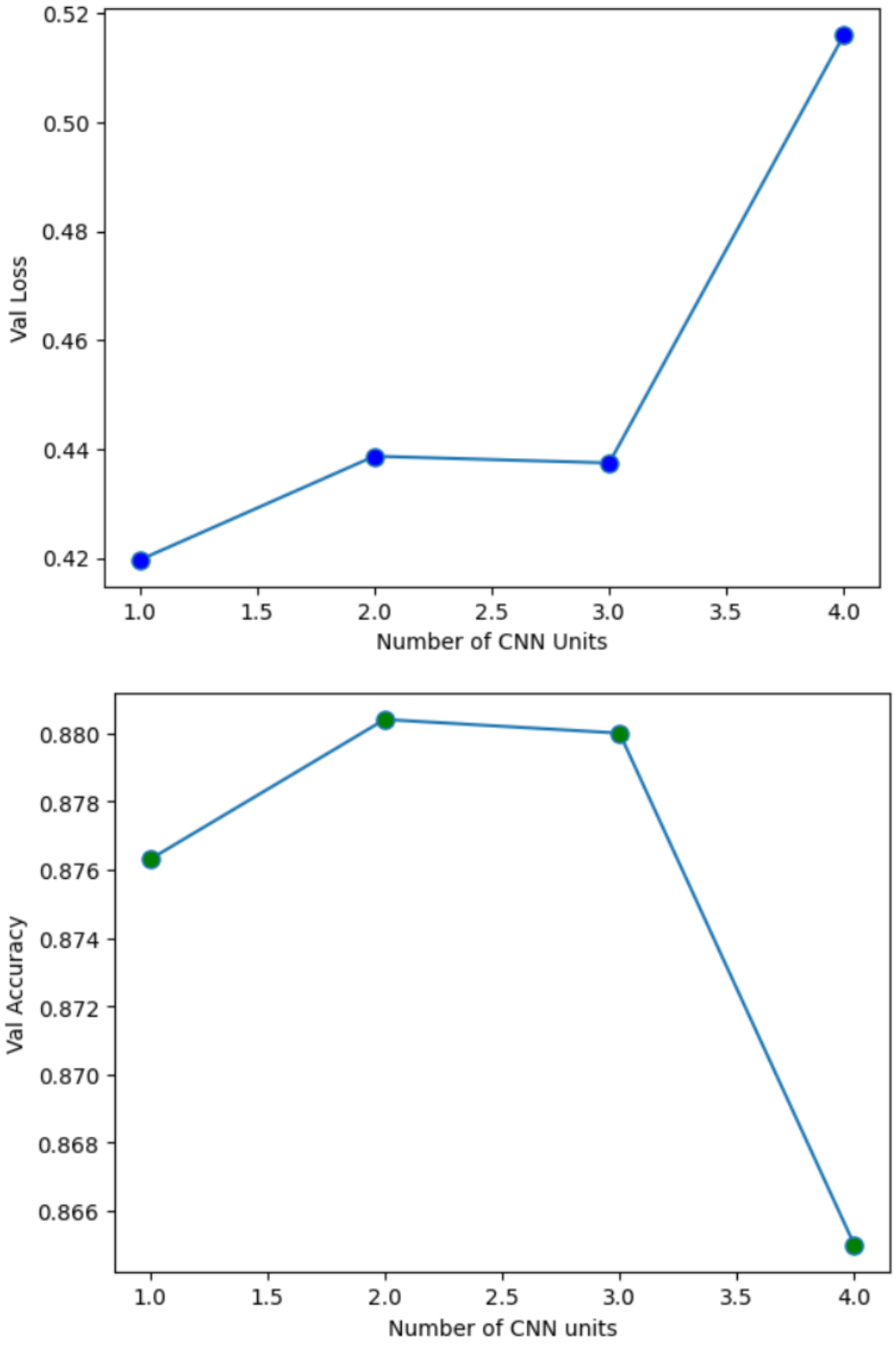
From validation loss graph, we notice that as the number of layers increase, the performance degrades.

2- Tune number of units in LSTM layer:
Values tried for units per lstm layer = [64, 32, 16]

The other hyperparameters are constant and are:

1- Learning rate = 0.001
2- Batch size= 128
3- Number of layers 1 (best value from above)
4- Activation function in lstm layer is tanh
5- Regulizer L2 = 0.02
6- One dense layer of 3 units with softmax activation

Values in order [64, 32, 32, 16] (32 is duplicated)



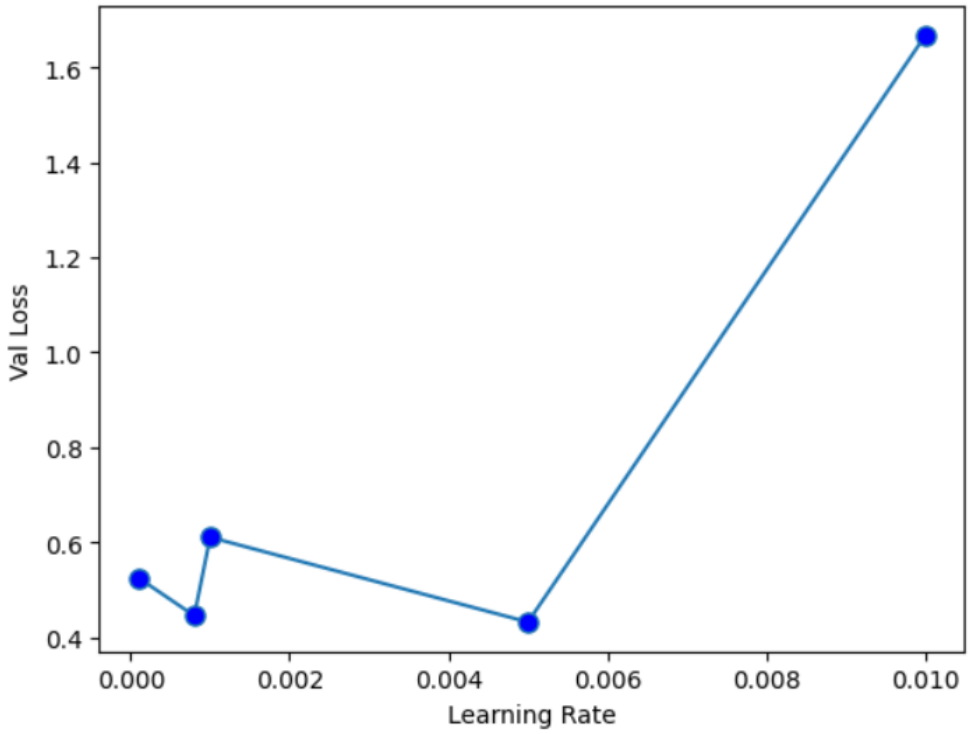
From validation loss graph, the larger the number of units is the best

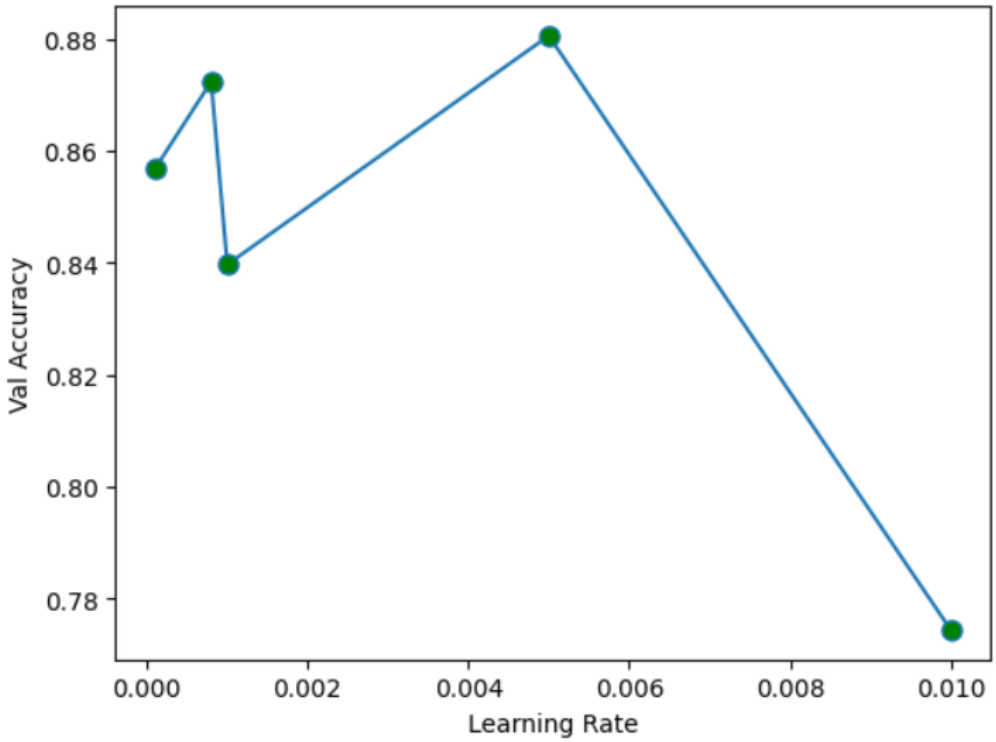
3- Tune learning rate:

Values tried for learning rates = [0.0001, 0.0008, 0.001, 0.005, 0.01]

The other hyperparameters are constant and are:

1- Number of lstm layers=1
2- Batch size= 128
3- Number of units in lstm layer is 64
4- Activation function in lstm layer is tanh
5- Regulizer L2 = 0.02
6- One dense layer of 3 units with softmax activation





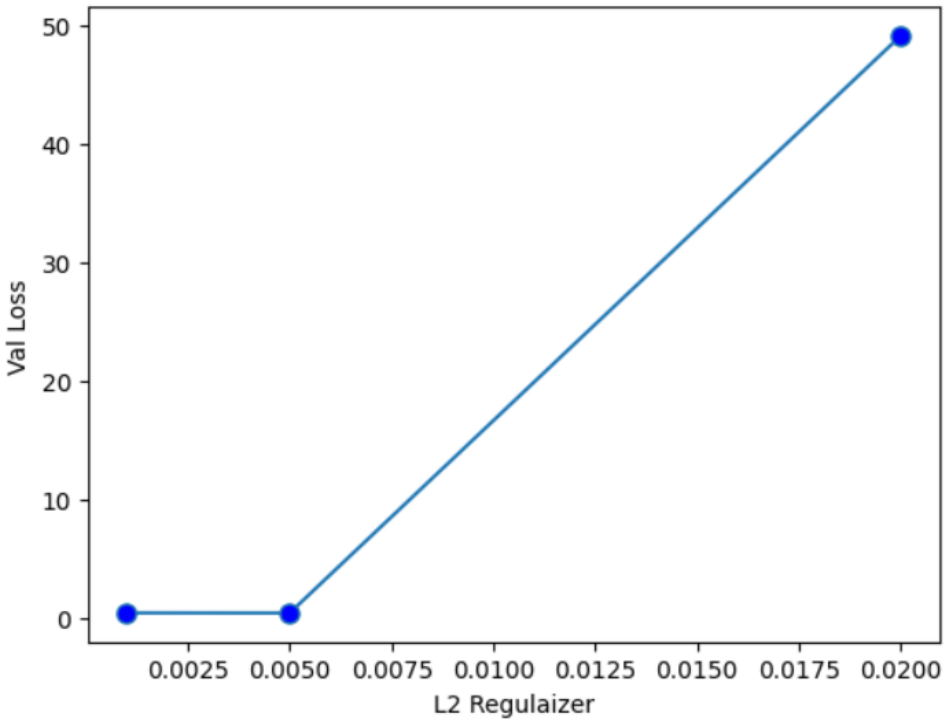
Best value is 0.005, however in other trials 0.001 yielded best performance. So, we tried in the last models, both of the values.

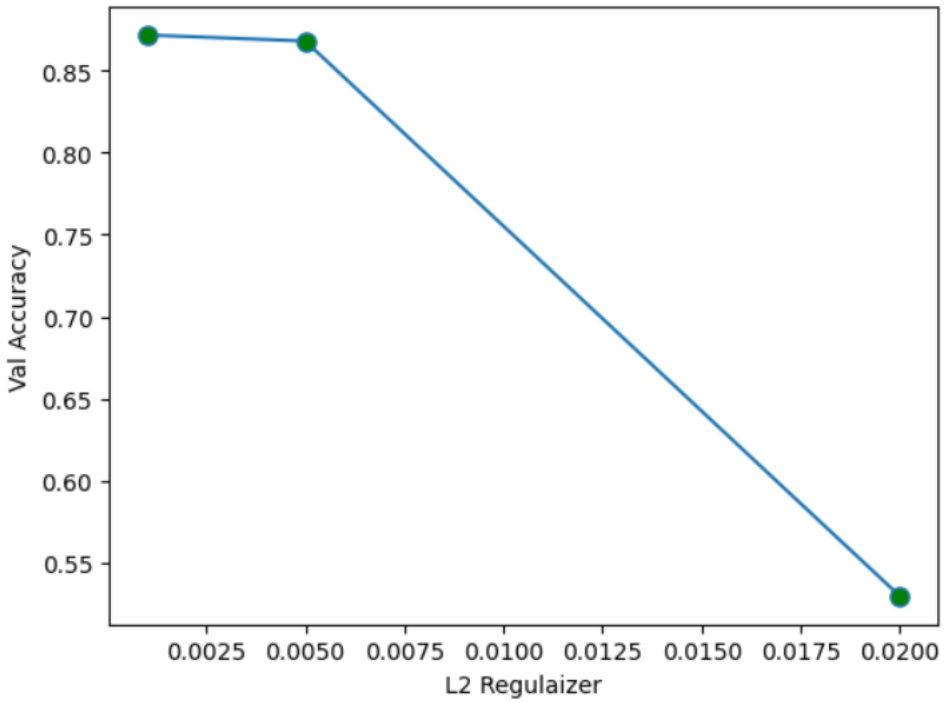
4- Tune L2:

Values tried for L2 = [0.0001, 0.005, 0.02]

The other hyperparameters are constant and are:

1- Number of lstm layers=1
2- Batch size= 128
3- Number of units in lstm layer is 64
4- Activation function in lstm layer is tanh
5- Learning Rate = 0.001
6- One dense layer of 3 units with softmax activation





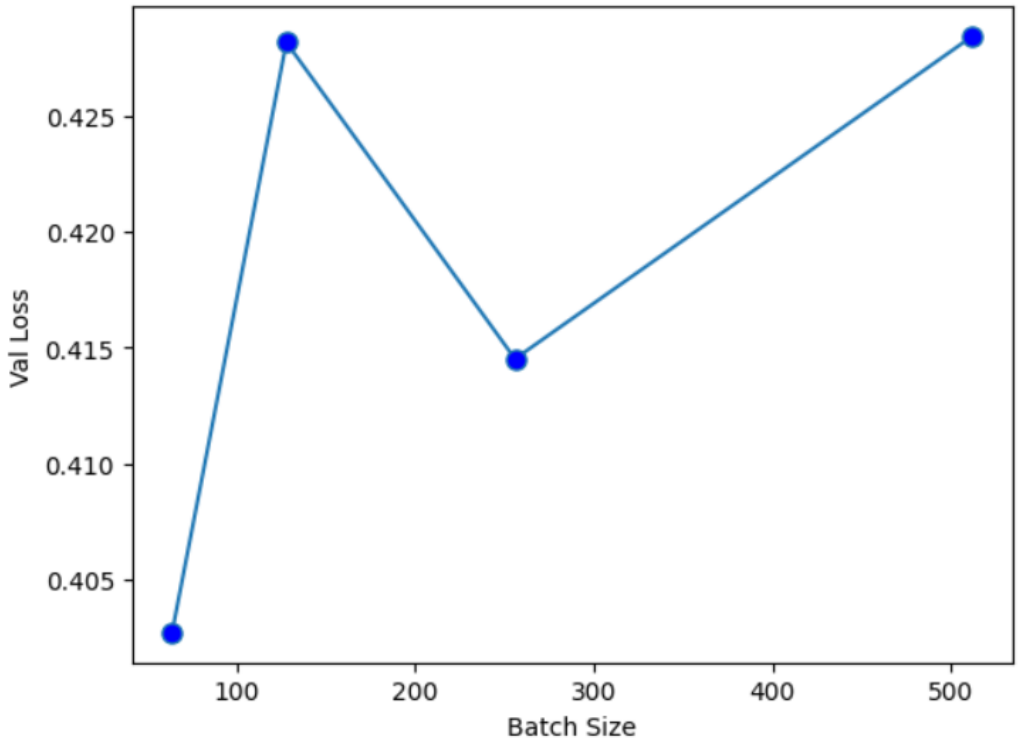
From validation loss graph, L2 = 0.005 is the best

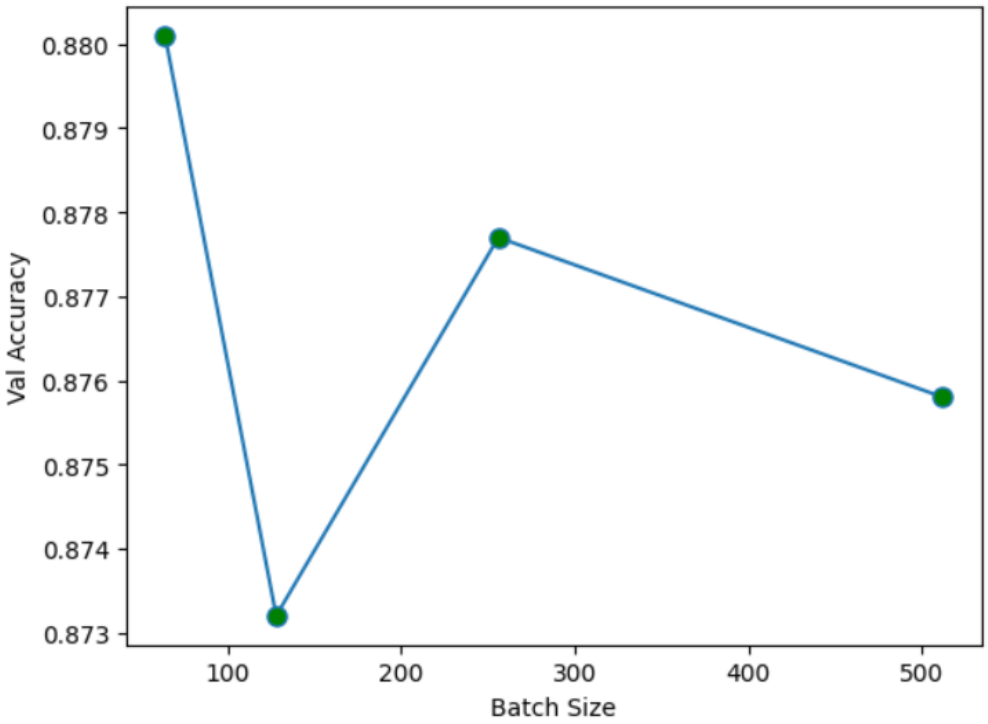
5- Tune batch size:

Values tried for batch size = [64, 128, 256, 512]

The other hyperparameters are constant and are:

1- Number of lstm layers=1
2- Learning rate = 0.005
3- Number of units in lstm layer is 64
4- Activation function in lstm layer is tanh
5- Regulizer L2 = 0.005
6- One dense layer of 3 units with softmax activation





From validation loss graph, the smaller the batch size the better, however slower training.

Using these results, we constructed 4 models, with small differences in parameters and chose the best by test accuracy.

All have batch size = 64 and trained for 5 epochs

Model 1:
1 LSTM hidden layer with 64 neurons
Learning rate 0.005
L2 0.005

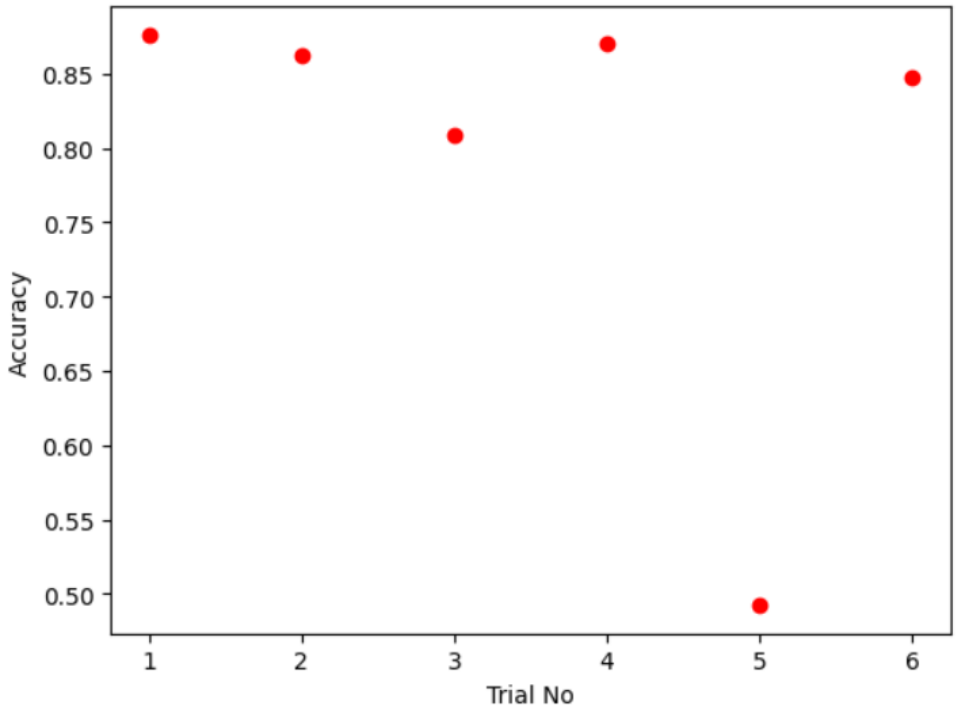
Model 2:
1 LSTM hidden layer with 32 neurons
Learning rate 0.005
L2 0.005

Model 3:
2 LSTM hidden layer with 32, 64 neurons
Learning rate 0.005
L2 0.02

Model 4:
1 LSTM hidden layer with 32 neurons
Learning rate 0.001
L2 0.02

Model 5:
4 LSTM hidden layer with 64, 32, 16, 8 neurons
Learning rate 0.001
L2 0.02

Model 6:
3 LSTM hidden layer with 32, 32, 64 neurons
Learning rate 0.001
L2 0.009

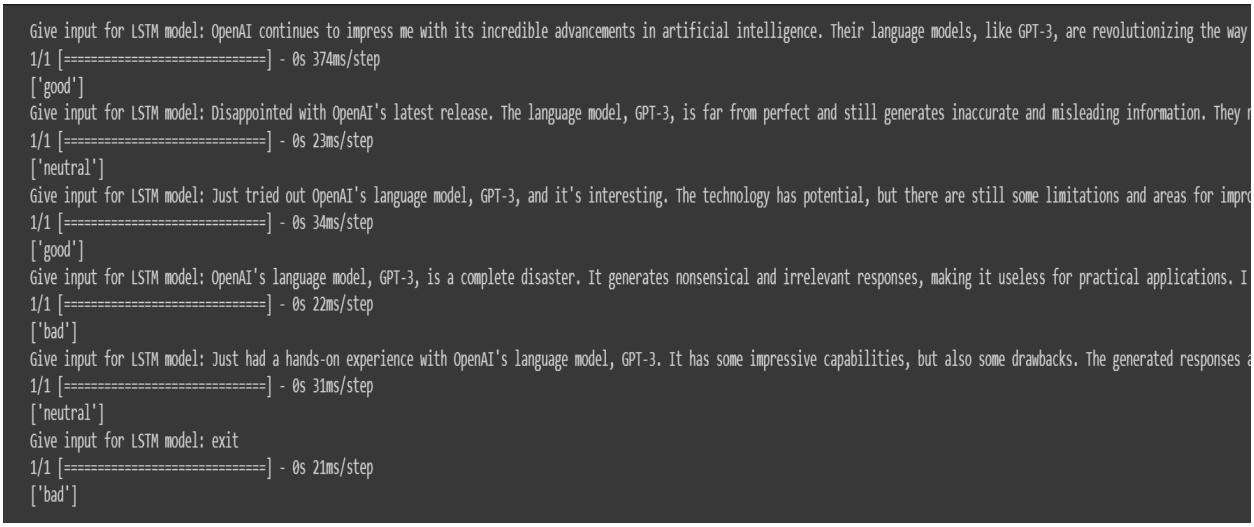


Best model is model 1 with accuracy 87.623

```
print("Best Test Accuarcy: ", max(accuracies)*100, " which is model ", (accuracies.index(max(accuracies)) + 1))
```

```
Best Test Accuarcy:  87.62369751930237  which is model  1
```

Taking inputs from user:



For clarity, the inputs are:

1- OpenAI continues to impress me with its incredible advancements in artificial intelligence. Their language models, like GPT-3, are revolutionizing the way we interact with technology. Exciting times ahead! #OpenAI #AI #Technology

2- Disappointed with OpenAI's latest release. The language model, GPT-3, is far from perfect and still generates inaccurate and misleading information. They need to improve their algorithms and provide better quality control. #OpenAI #GPT3 #Disappointed

3- Just tried out OpenAI's language model, GPT-3, and it's interesting. The technology has potential, but there are still some limitations and areas for improvement. Looking forward to seeing how OpenAI continues to develop their AI capabilities. #OpenAI #GPT3 #AI

4- OpenAI's language model, GPT-3, is a complete disaster. It generates nonsensical and irrelevant responses, making it useless for practical applications. I expected much more from such a hyped-up AI project. #OpenAI #GPT3 #Disappointed

5- Just had a hands-on experience with OpenAI's language model, GPT-3. It has some impressive capabilities, but also some drawbacks. The generated responses are sometimes accurate, but other times miss the mark. It's a mixed bag overall. #OpenAI #GPT3 #AI

Outputs by model are: good, neutral, good, bad, neutral, bad

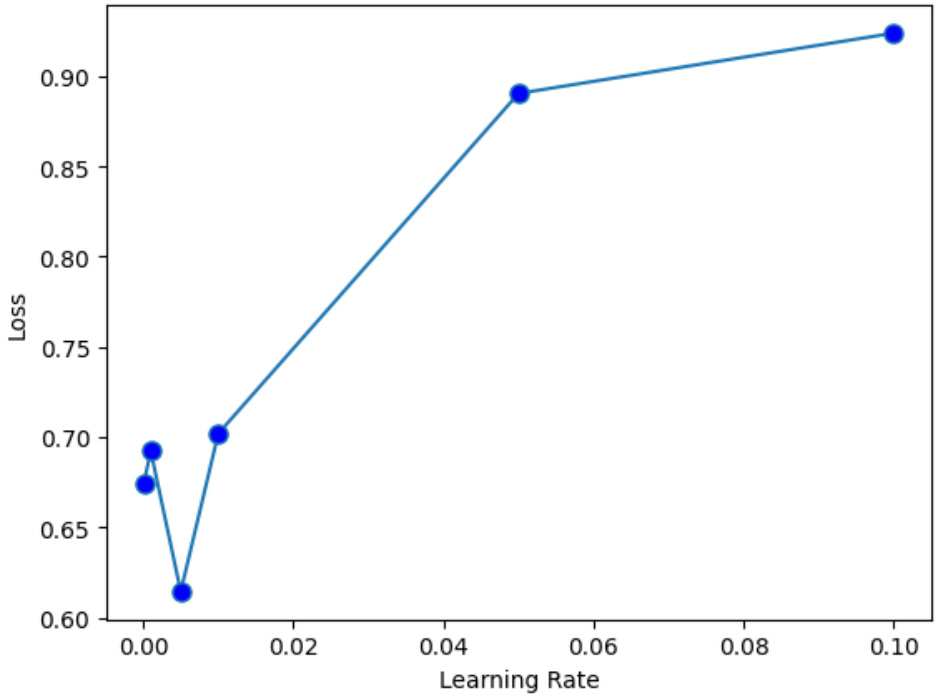
CNN

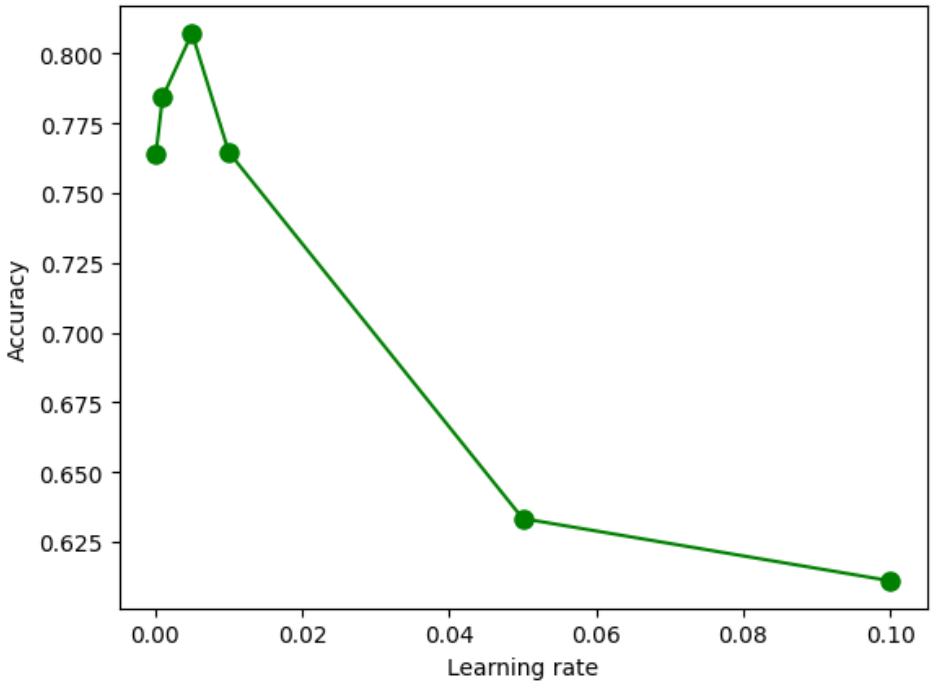
*[All graphs, during hyperparameter tuning, are on the validation **loss** and **accuracy**]*

- **Weights of embedding layer are not trainable (Trainable = False)**

1- Tune Learning Rate value:

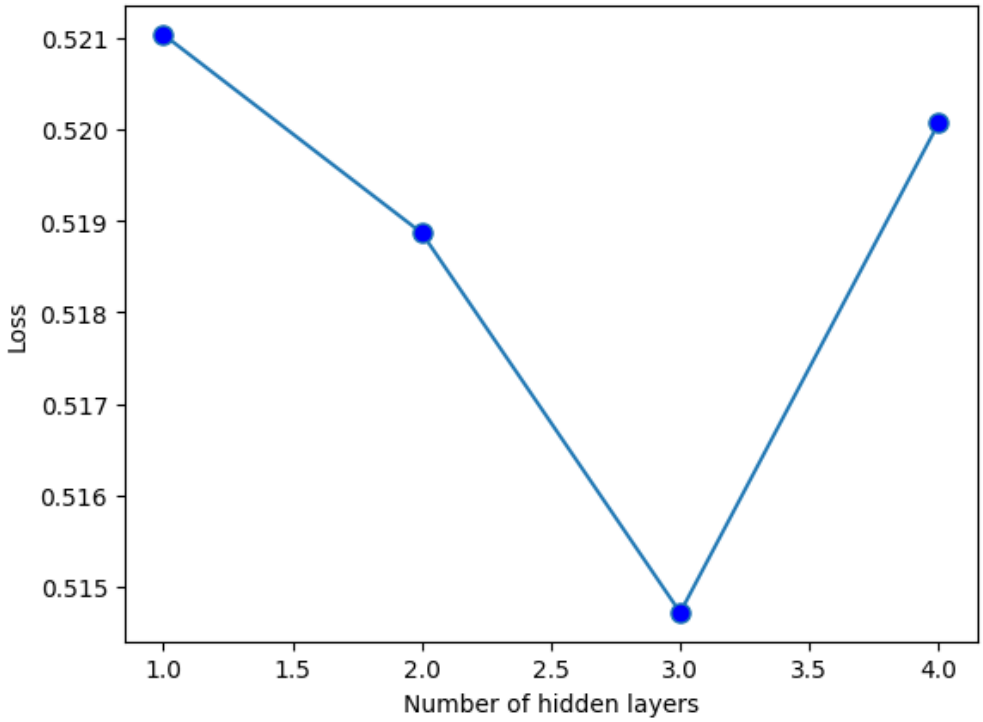
Values tried for learning rate = [0.0001, 0.001, 0.005, 0.01, 0.05, 0.1]
The other hyperparameters were constant with values:
1.Number of Conv1D layers = 1
2.Number of Filters = 128
3.Number of Dense layer units = 128
4.Regulizer L2 value = 0.0003
5.Batch size = 128

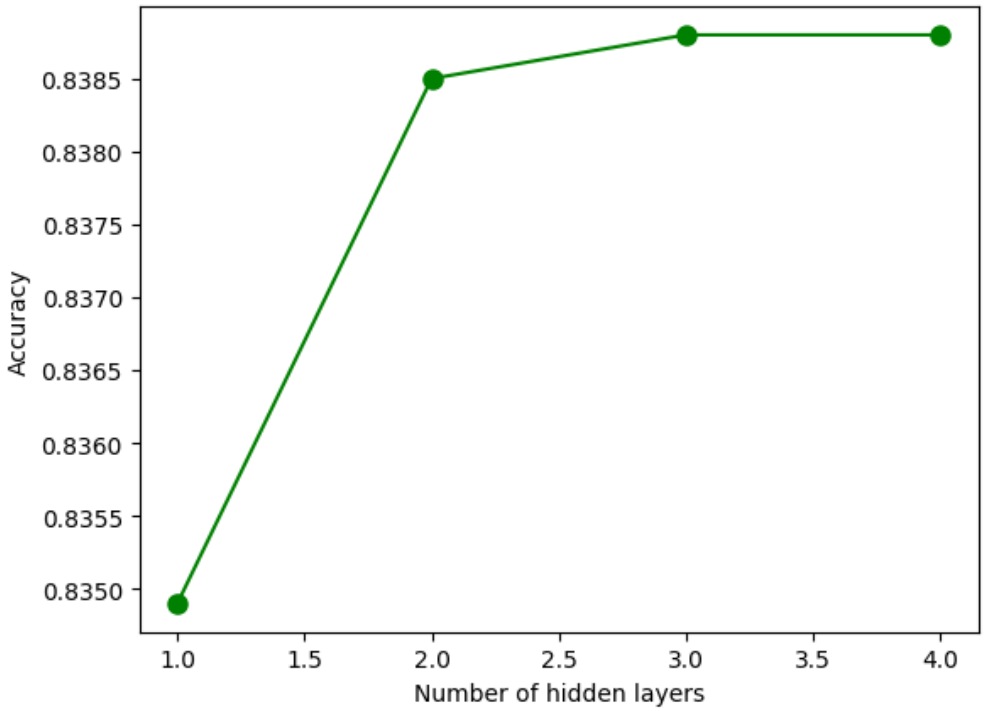




We notice as the learning rate increases, the performance degrades.

2- **Tune number of Conv1D layers:**
Values tried for number of Conv1D layers = [1, 2, 3, 4]
The other hyperparameters were constant with values:
1. Learning rate = 0.005 *(best learning rate value from above)*
2. Number of Filters = 128
3. Number of Dense layer units = 128
4. Regulizer L2 value = 0.0003
5. Batch size = 128





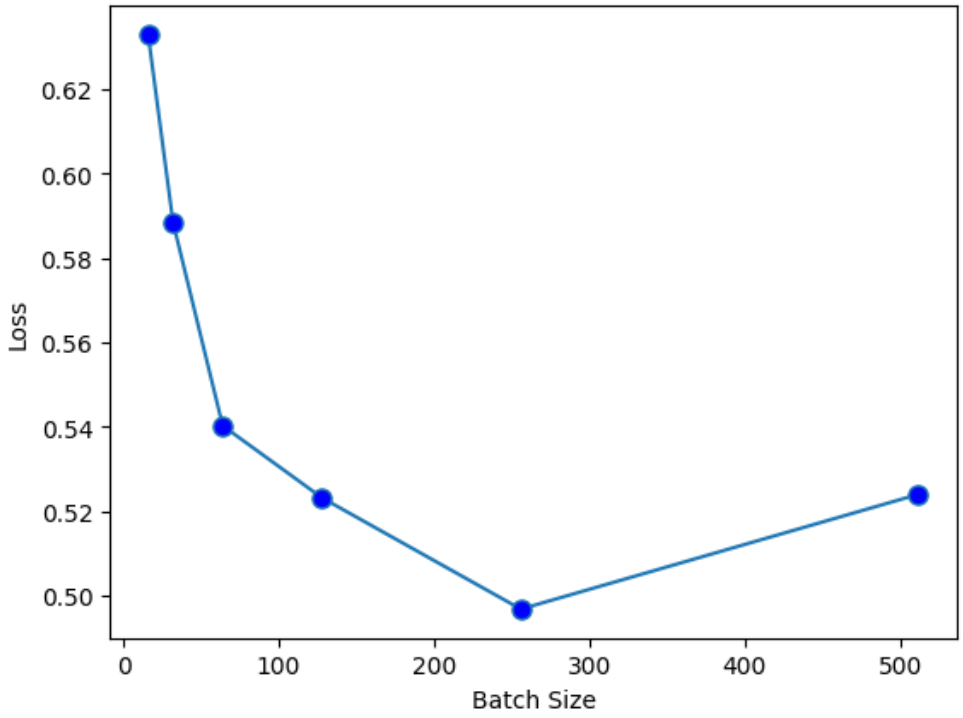
The best number of Conv1D layers is 3.

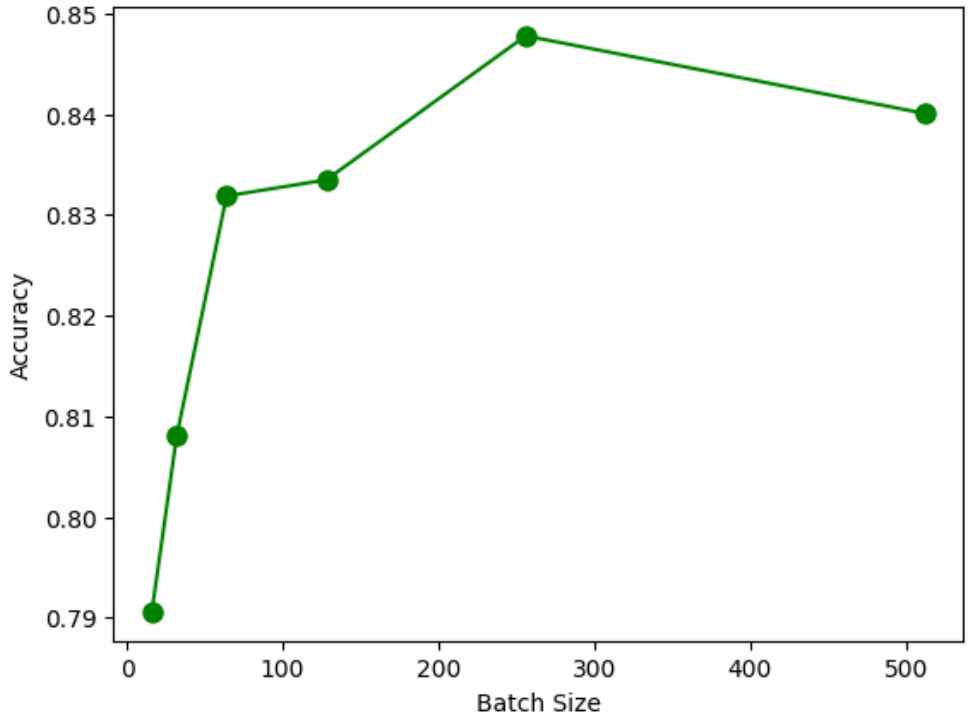
3- Tune batch size:

Values tried for batch size = [16, 32, 64, 128, 256, 512]

The other hyperparameters were constant with values:

1. Learning rate = 0.005 *(best learning rate value from above)*
2. Number of Conv1D layers = 3 *(best number of Conv1D layers from above)*
3. Number of Filters = 128
4. Number of Dense layer units = 128
5. Regulizer L2 value = 0.0003



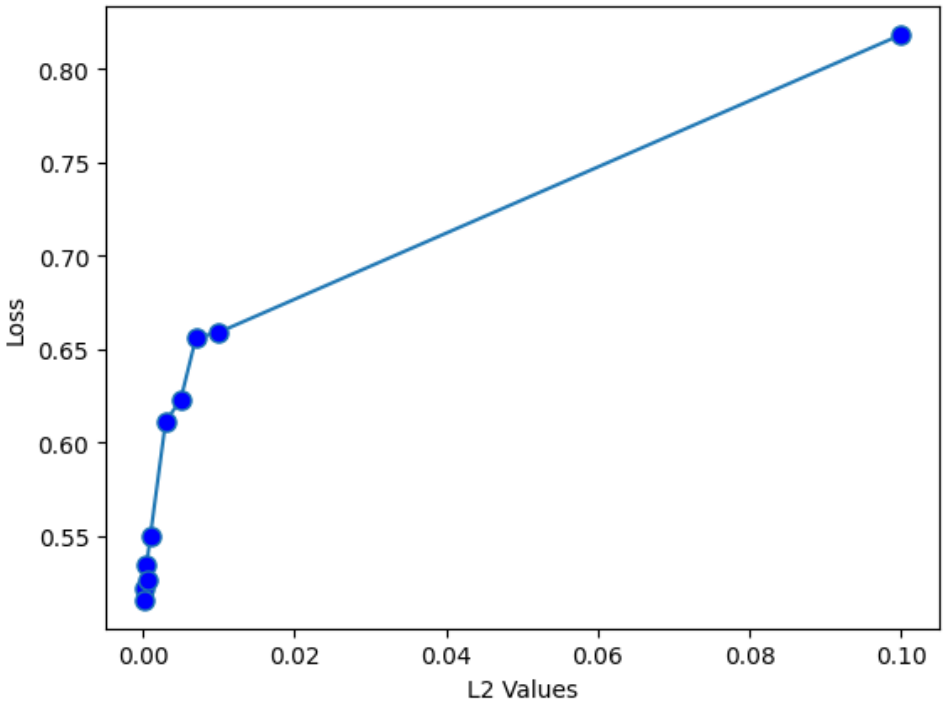


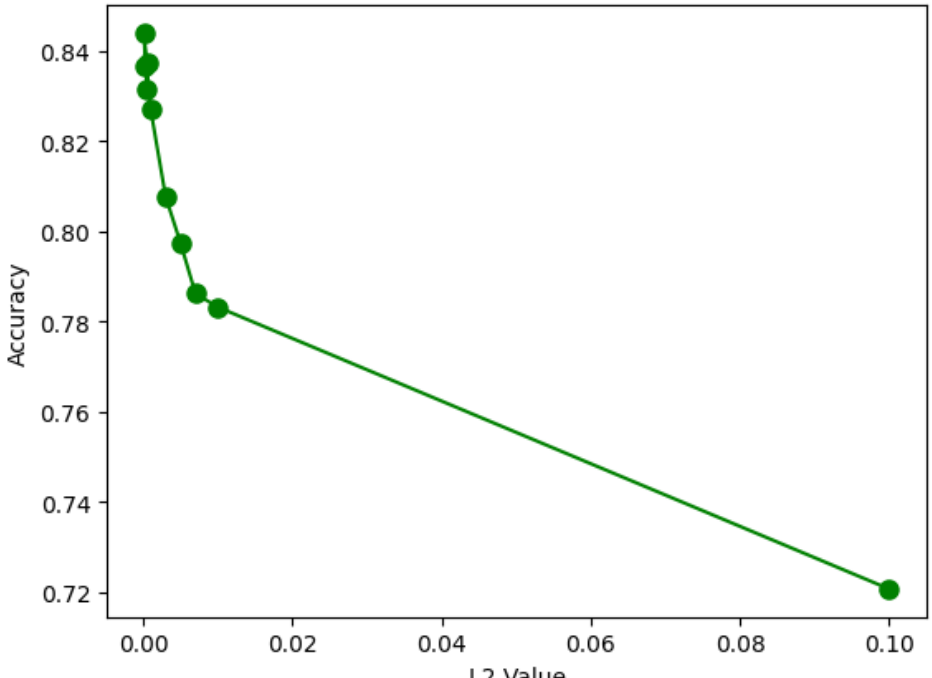
The best batch size value is 256.

4- Tune L2 value:

Values tried for L2 value = [0.1, 0.01, 0.007, 0.005, 0.003, 0.001, 0.0001, 0.0003, 0.0005, 0.0007]

1. Learning rate = 0.005 *(best learning rate value from above)*
2. Number of Conv1D layers = 3 *(best number of Conv1D layers from above)*
3. Number of Filters = 128
4. Number of Dense layer units = 128
5. Batch size = 256 *(best batch size value from above)*



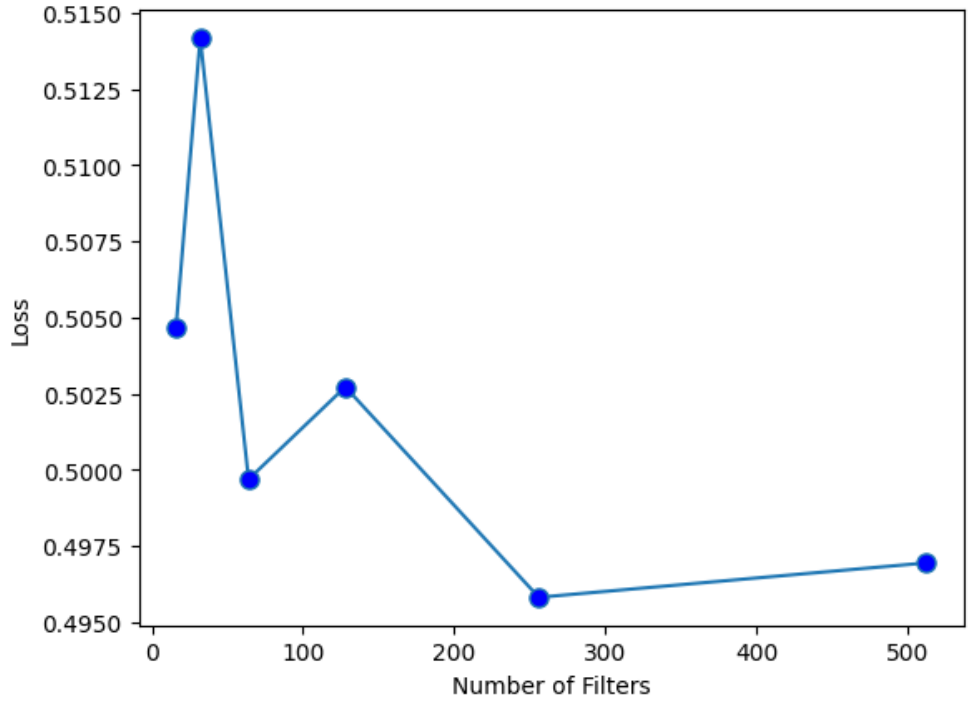


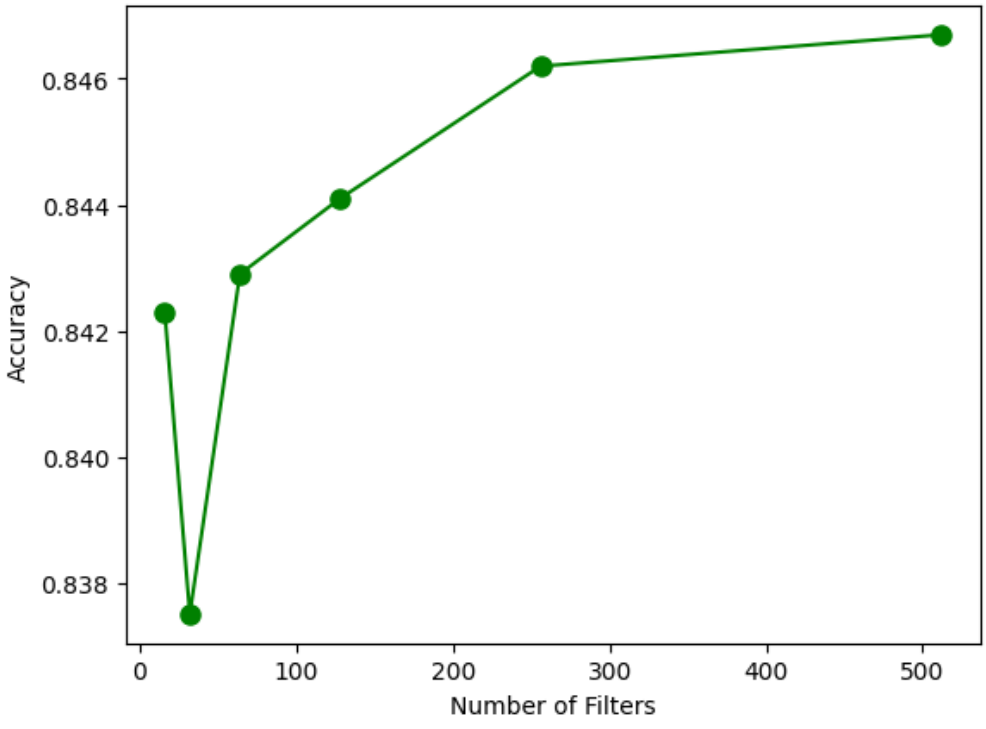
The best L2 value is 0.0003.

5- Tune Number of filters at Conv1D layer/s:

Values tried for number of filters = [16, 32, 64, 128, 256, 512]

1. Learning rate = 0.005 *(best learning rate value from above)*
2. Number of Conv1D layers = 3 *(best number of Conv1D layers from above)*
3. Regulizer L2 value = 0.0003 *(best L2 value from above)*
4. Number of Dense layer units = 128
5. Batch size = 256 *(best batch size value from above)*



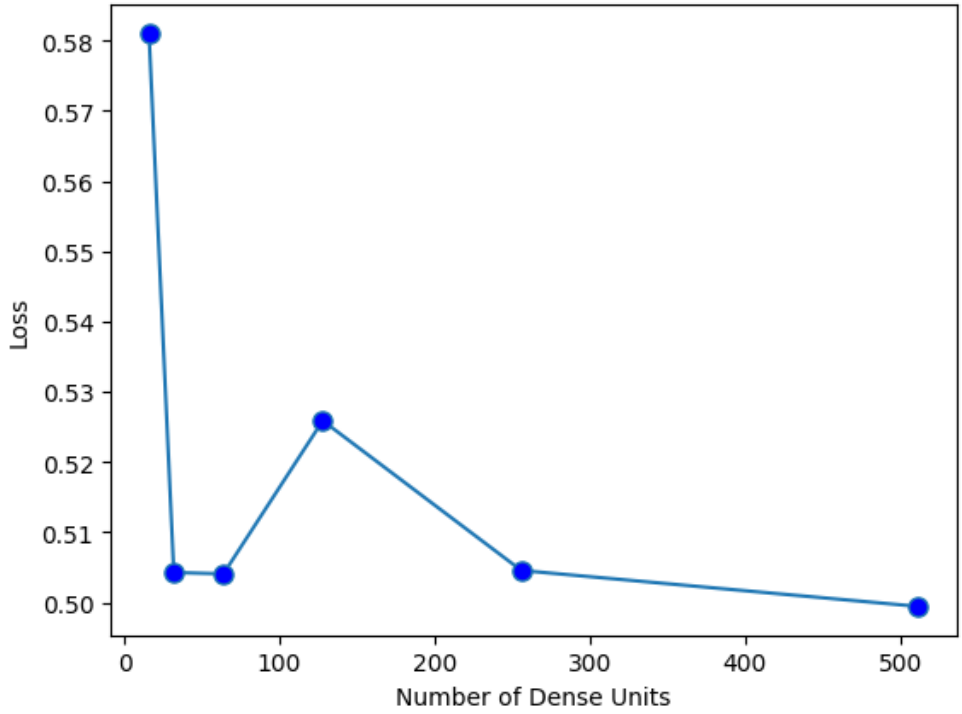


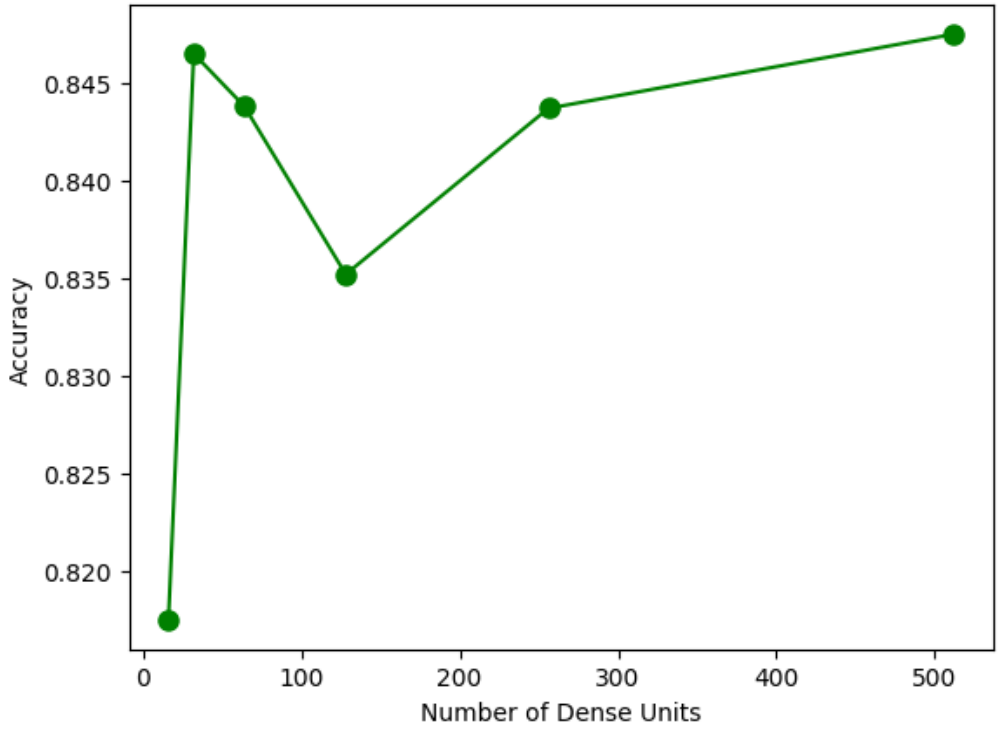
The best number of filters in Conv1D layers is 256.

6- Tune Number of units of Dense layer:

Values tried for number of units = [16, 32, 64, 128, 256, 512]

1. Learning rate = 0.005 *(best learning rate value from above)*
2. Number of Conv1D layers = 3 *(best number of Conv1D layers from above)*
3. Regulizer L2 value = 0.0003 *(best L2 value from above)*
4. Number of Filters = 256 *(best number of filters from above)*
5. Batch size = 256 *(best batch size value from above)*



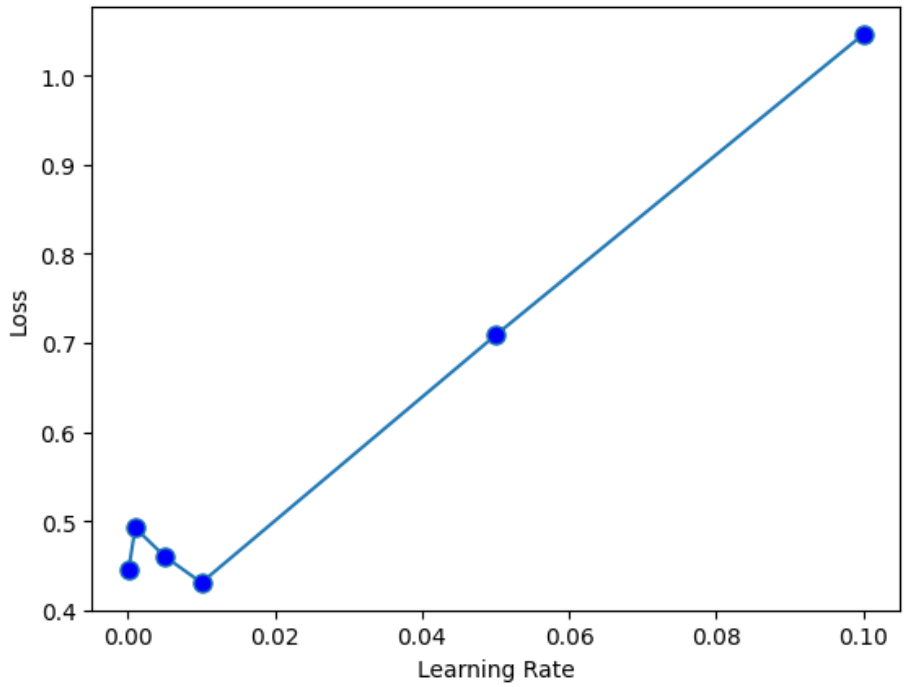


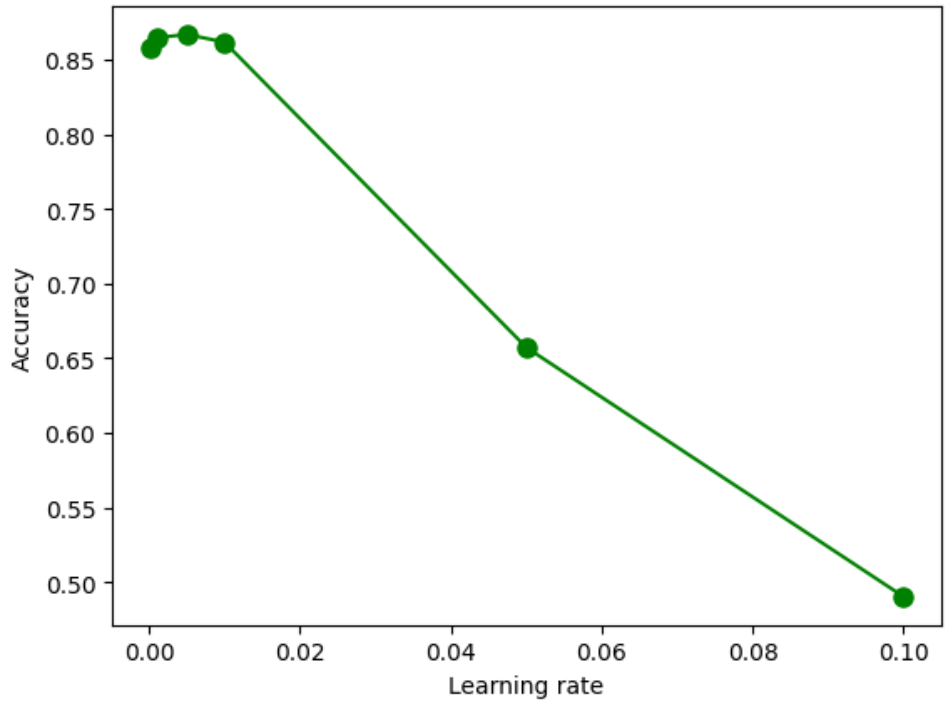
The best number of units of the dense layer is 512.

- **Weights of embedding layer are trainable (Trainable = True)**

1- Tune Learning Rate value:
Values tried for learning rate = [0.0001, 0.001, 0.005, 0.01, 0.05, 0.1]
The other hyperparameters were constant with values:

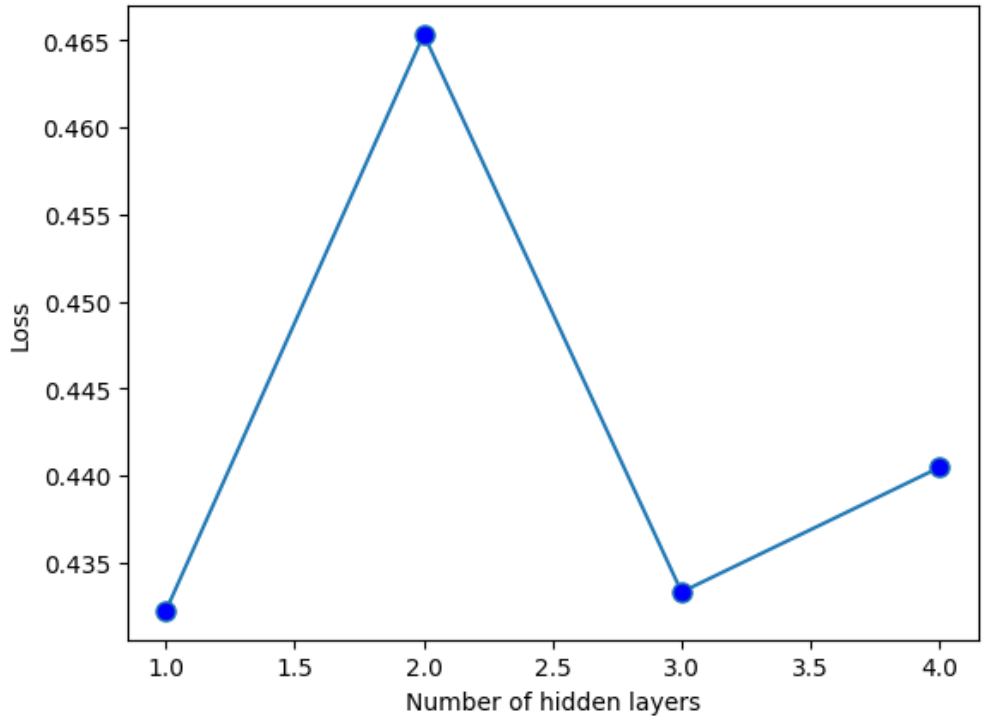
1. Number of Conv1D layers = 1
2. Number of Filters = 128
3. Number of Dense layer units = 128
4. Regulizer L2 value = 0.0003
5. Batch size = 128

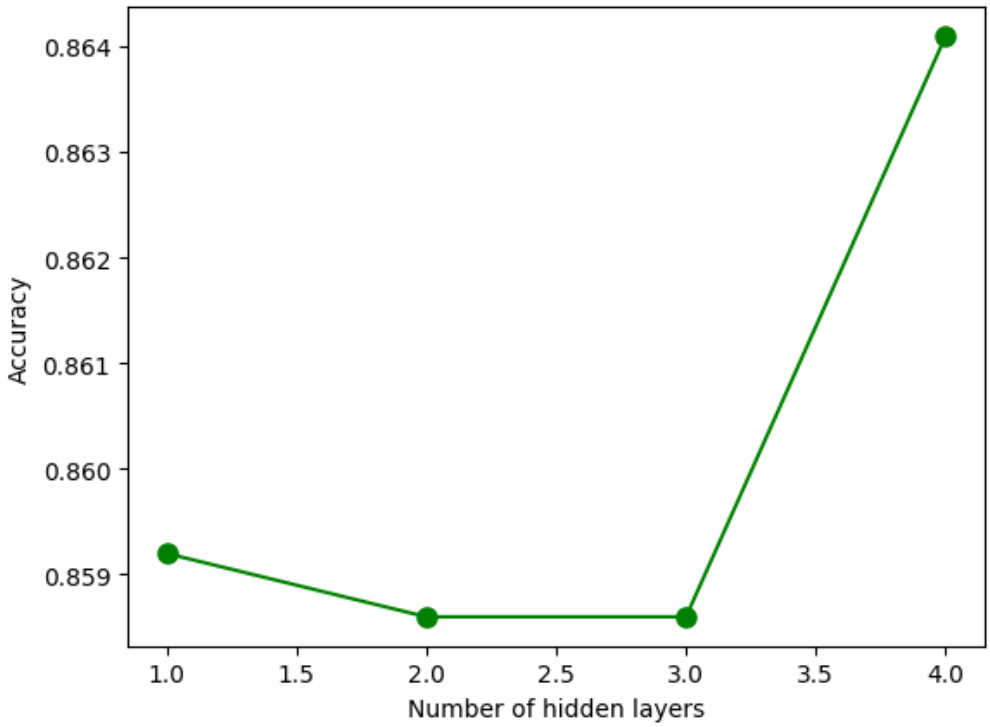




The best learning rate is 0.01.

2- **Tune number of Conv1D layers:**
Values tried for number of Conv1D layers = [1, 2, 3, 4]
The other hyperparameters were constant with values:
1. Learning rate = 0.01 *(best learning rate value from above)*
2. Number of Filters = 128
3. Number of Dense layer units = 128
4. Regulizer L2 value = 0.0003
5. Batch size = 128





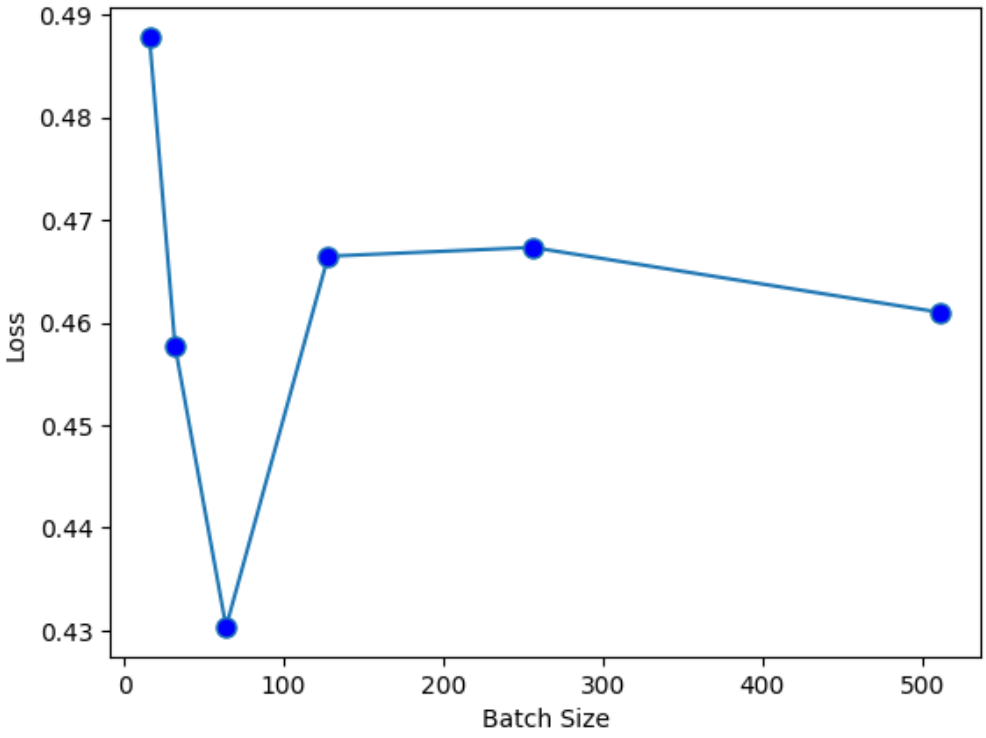
The best number of Conv1D layers is 1.

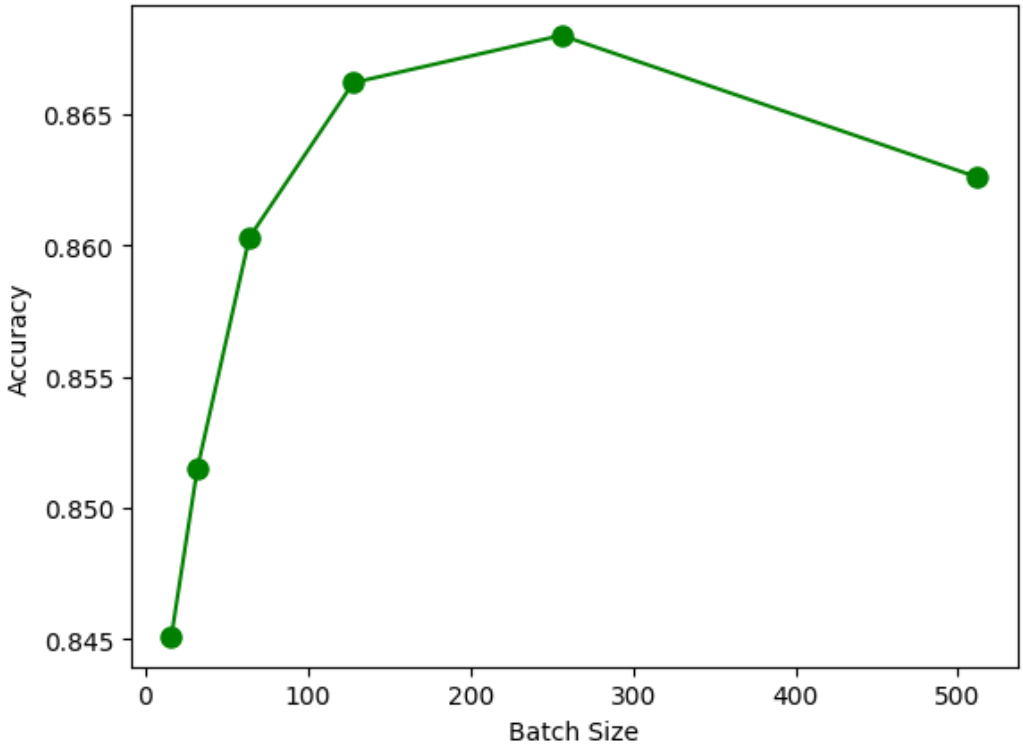
3- **Tune batch size:**

Values tried for batch size = [16, 32, 64, 128, 256, 512]

The other hyperparameters were constant with values:

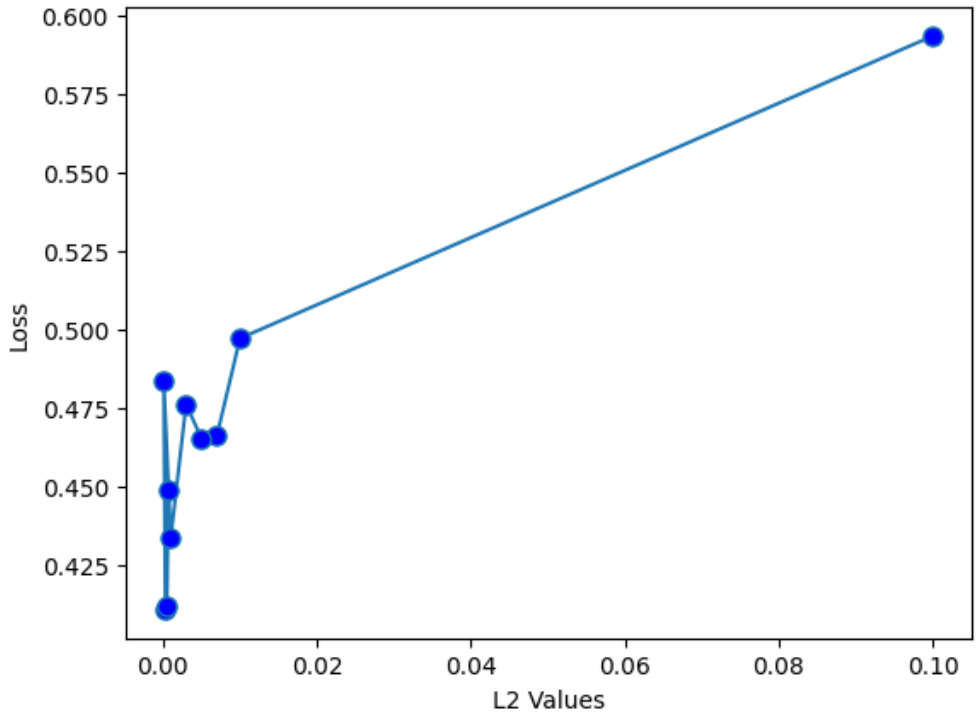
1. Learning rate = 0.01 *(best learning rate value from above)*
2. Number of Conv1D layers = 1 *(best number of Conv1D layers from above)*
3. Number of Filters = 128
4. Number of Dense layer units = 128
5. Regulizer L2 value = 0.0003

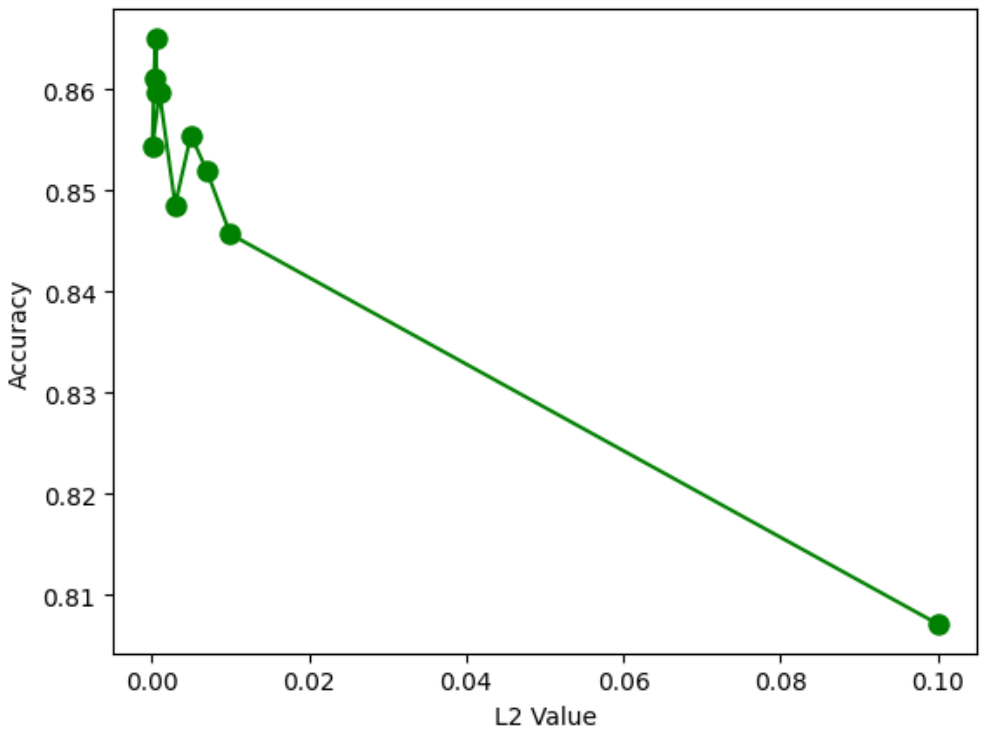




The best batch size value is 64.

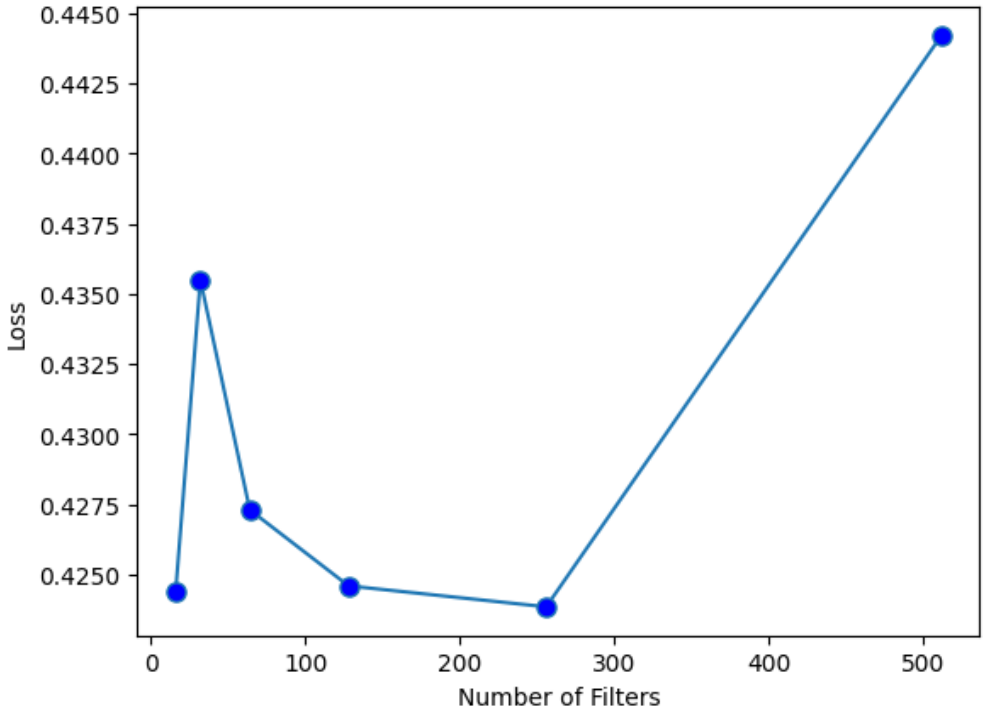
4- **Tune L2 value:**
   Values tried for L2 value = [0.1, 0.01, 0.007, 0.005, 0.003, 0.001, 0.0001, 0.0003, 0.0005, 0.0007]
   1. Learning rate = 0.01 *(best learning rate value from above)*
   2. Number of Conv1D layers = 1 *(best number of Conv1D layers from above)*
   3. Number of Filters = 128
   4. Number of Dense layer units = 128
   5. Batch size = 64 *(best batch size value from above)*

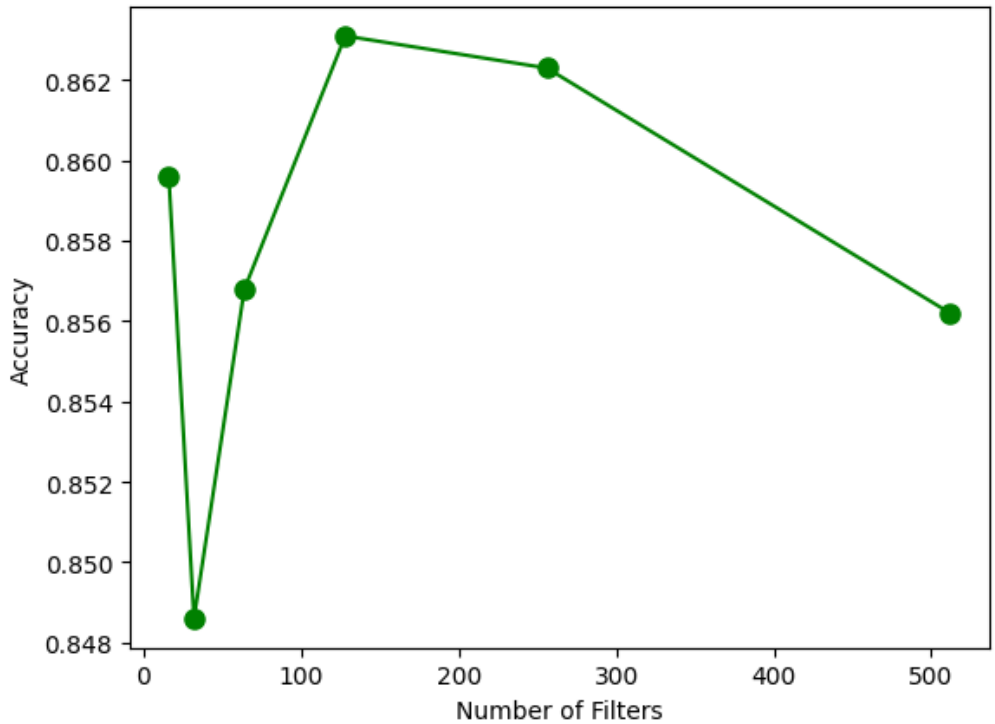




The best L2 value is 0.0003.

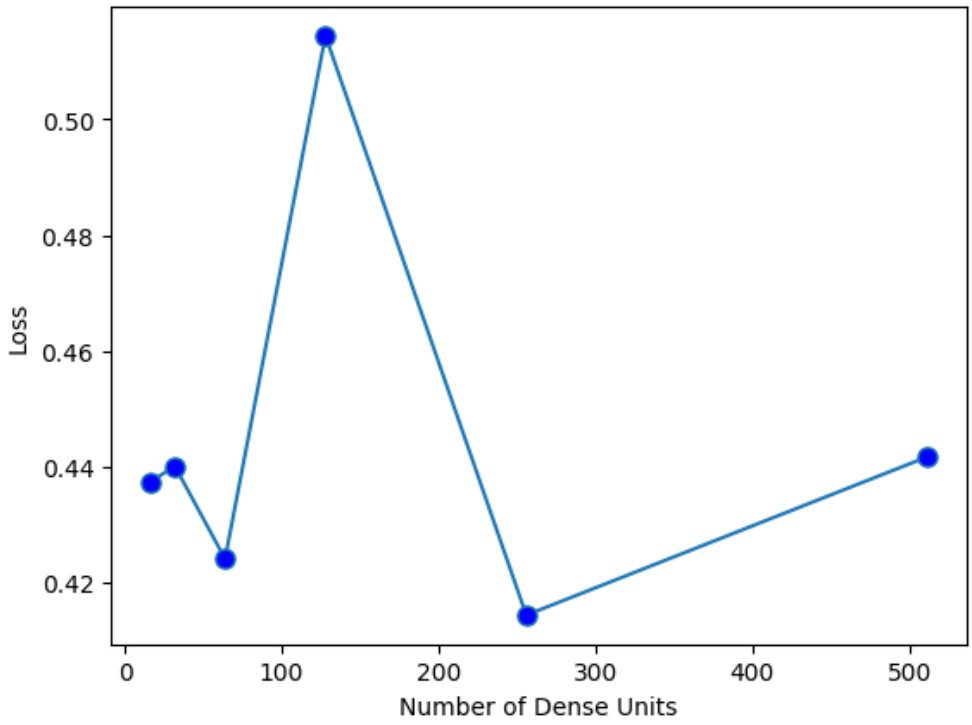
5- **Tune Number of filters at Conv1D layer/s:**
Values tried for number of filters = [16, 32, 64, 128, 256, 512]
1. Learning rate = 0.01 *(best learning rate value from above)*
2. Number of Conv1D layers = 1 *(best number of Conv1D layers from above)*
3. Regulizer L2 value = 0.0003 *(best L2 value from above)*
4. Number of Dense layer units = 128
5. Batch size = 64 *(best batch size value from above)*

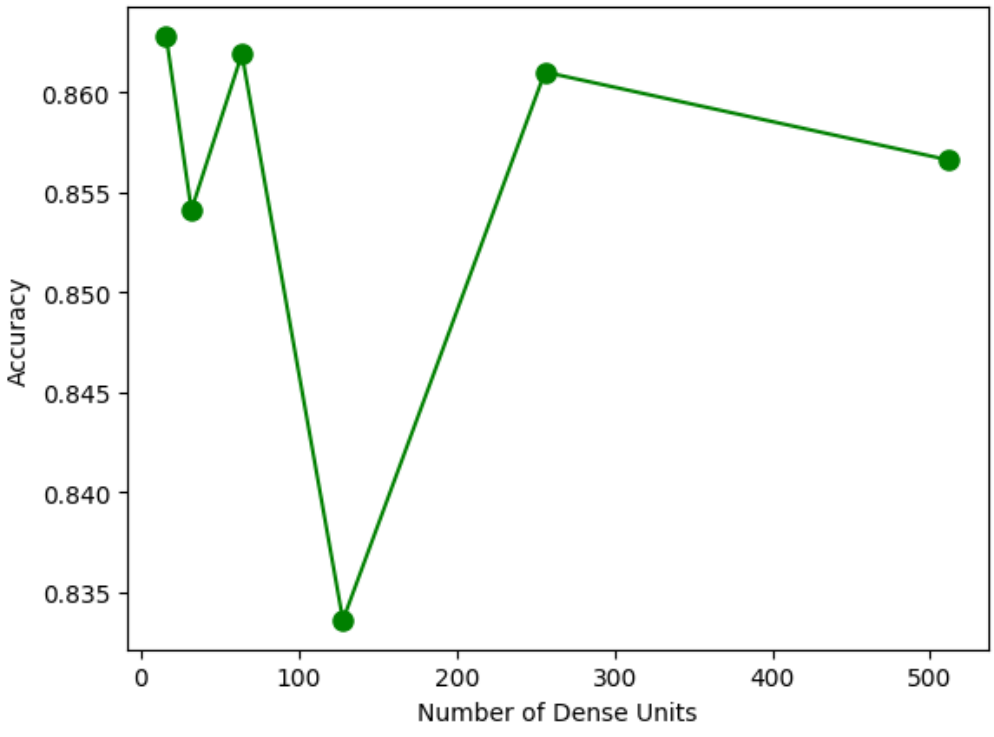




The best number of filters is 256.

6- **Tune Number of units of Dense layer:**
   Values tried for number of units = [16, 32, 64, 128, 256, 512]
   1. Learning rate = 0.01 *(best learning rate value from above)*
   2. Number of Conv1D layers = 1 *(best number of Conv1D layers from above)*
   3. Regulizer L2 value = 0.0003 *(best L2 value from above)*
   4. Number of Filters = 256 *(best number of filters from above)*
   5. Batch size = 64 *(best batch size value from above)*





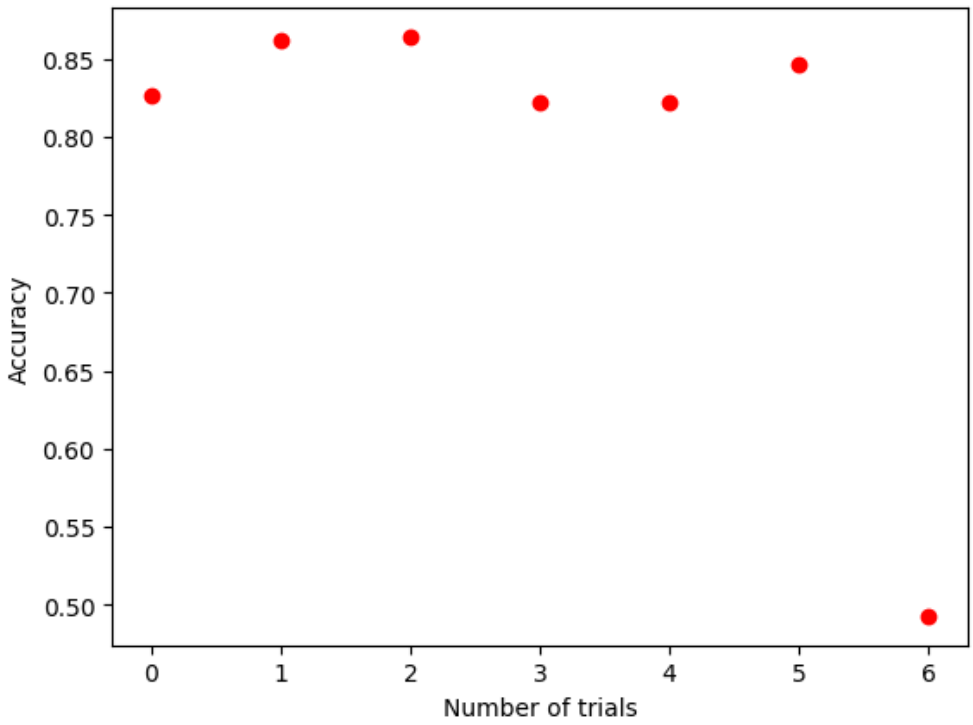
The best number of dense units is 256.

#Note:

All hyperparameters are chosen based on the minimum validation loss, as it's an indication that the model works well with more generalization.

Using these results, we constructed 7 models, with differences in parameters and chose the best by test accuracy.

- **Trials:**
  1. **Model 0**
     Learning rate: 0.005, Conv layers: 3
     L2 value: 0.0003, Filters: 256
     Dense units: 512, Batch size: 256, trainable = False
  2. **Model 1**
     Learning rate: 0.01, Conv layers: 1
     L2 value: 0.0003, Filters: 256
     Dense units: 256, Batch size: 5, trainable = True
  3. **Model 2**
     Learning rate: 0.005, Conv layers: 3
     L2 value: 0.0003, Filters: 256
     Dense units: 512, Batch size: 256, trainable = True
  4. **Model 3**
     Learning rate: 0.005, Conv layers: 2
     L2 value: 0.0003, Filters: 64
     Dense units: 128, Batch size: 128, trainable = False
  5. **Model 4**
     Learning rate: 0.005, Conv layers: 4
     L2 value: 0.0003, Filters: 64
     Dense units: 128, Batch size: 128, trainable = False
  6. **Model 5**
     Learning rate: 0.01, Conv layers: 2
     L2 value: 0.007, Filters: 64
     Dense units: 128, Batch size: 128, trainable = True
  7. **Model 6**
     Learning rate: 0.05, Conv layers: 4
     L2 value: 0.003, Filters: 64
     Dense units: 128, Batch size: 128, trainable = True



As shown in the plot, there are 2 models that have accuracies very close to each other (models 1 and 2) so we choose model 1 which has a smaller loss.

With accuracy 86.2077 and loss 0.4379684

➔ Model with best accuracy:

```
cnnModel1 = defineCNNModel (convLayersTrainable, l2Trainable, numFiltersTrainable, denseUnitsTrainable,  trainableBool = True)
optimizer = tf.keras.optimizers.Adam(learning_rate=learningRateTrainable)
cnnModel1.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
cnnModel1.fit(x_train, yTrain, epochs=numEpochs, batch_size=batchSizeTrianable, validation_data=(x_val, y_val))
```

```
Epoch 1/5
2193/2193 [==============================] - 99s 44ms/step - loss: 0.5682 - accuracy: 0.8165 - val_loss: 0.4462 - val_accuracy: 0.8498
Epoch 2/5
2193/2193 [==============================] - 64s 29ms/step - loss: 0.4012 - accuracy: 0.8659 - val_loss: 0.4093 - val_accuracy: 0.8603
Epoch 3/5
2193/2193 [==============================] - 56s 25ms/step - loss: 0.3625 - accuracy: 0.8801 - val_loss: 0.4052 - val_accuracy: 0.8634
Epoch 4/5
2193/2193 [==============================] - 54s 25ms/step - loss: 0.3339 - accuracy: 0.8922 - val_loss: 0.4201 - val_accuracy: 0.8647
Epoch 5/5
2193/2193 [==============================] - 54s 24ms/step - loss: 0.3121 - accuracy: 0.9017 - val_loss: 0.4288 - val_accuracy: 0.8646
<keras.callbacks.History at 0x7ff50bc9d720>
```

```
cnnModelLoss1, cnnModelAccuracy1 = cnnModel1.evaluate(x_test, yTest)
print(cnnModelLoss1, cnnModelAccuracy1)
```

```
1371/1371 [==============================] - 6s 4ms/step - loss: 0.4380 - accuracy: 0.8621
0.4379680752754211 4 0.8620775938034058
```

Taking input from user:

```
ChatGPT is amazing! It always knows the answers to my questions. #AI #chatbot #amazing
1/1 [==============================] - 0s 22ms/step
['good']
ChatGPT is helpful for basic questions, but it's not always reliable. #AI #chatbot #helpfulbutnotperfect
1/1 [==============================] - 0s 22ms/step
['neutral']
ChatGPT is terrible. It never understands what I'm asking and its responses are always wrong. #AI #chatbot #terrible
1/1 [==============================] - 0s 30ms/step
['bad']
I love using ChatGPT for research! It's like having a personal assistant that knows everything. #AI #chatbot #research
1/1 [==============================] - 0s 31ms/step
['good']
ChatGPT can be useful for quick answers, but it's not a substitute for human expertise. #AI #chatbot #usefulbutnotperfect
1/1 [==============================] - 0s 20ms/step
['neutral']
ChatGPT is a waste of time. Its responses are always irrelevant and it never understands what I'm asking. #AI #chatbot #wasteoftime
1/1 [==============================] - 0s 22ms/step
['bad']
OpenAI's language model, GPT-3, is a complete disaster. It generates nonsensical and irrelevant responses, making it useless for practical applications
1/1 [==============================] - 0s 23ms/step
['bad']
Just had a hands-on experience with OpenAI's language model, GPT-3. It has some impressive capabilities, but also some drawbacks. The generated respons
1/1 [==============================] - 0s 22ms/step
['neutral']
Just tried out OpenAI's language model, GPT-3, and it's interesting. The technology has potential, but there are still some limitations and areas for i
1/1 [==============================] - 0s 31ms/step
['good']
exit
```

For clarity, the inputs are:

1. ChatGPT is amazing! It always knows the answers to my questions. #AI #chatbot #amazing
2. ChatGPT is helpful for basic questions, but it's not always reliable. #AI #chatbot #helpfulbutnotperfect
3. ChatGPT is terrible. It never understands what I'm asking and its responses are always wrong. #AI #chatbot #terrible
4. I love using ChatGPT for research! It's like having a personal assistant that knows everything. #AI #chatbot #research
5. ChatGPT can be useful for quick answers, but it's not a substitute for human expertise. #AI #chatbot #usefulbutnotperfect
6. ChatGPT is a waste of time. Its responses are always irrelevant and it never understands what I'm asking. #AI #chatbot #wasteoftime
7. OpenAI's language model, GPT-3, is a complete disaster. It generates nonsensical and irrelevant responses, making it useless for practical applications. I expected much more from such a hyped-up AI project. #OpenAI #GPT3 #Disappointed
8. Just had a hands-on experience with OpenAI's language model, GPT-3. It has some impressive capabilities, but also some drawbacks. The generated responses are sometimes accurate, but other times miss the mark. It's a mixed bag overall. #OpenAI #GPT3 #AI
9. Just tried out OpenAI's language model, GPT-3, and it's interesting. The technology has potential, but there are still some limitations and areas for improvement. Looking forward to seeing how OpenAI continues to develop their AI capabilities. #OpenAI #GPT3 #AI

Outputs by model are good, neutral, bad, good, neutral, bad, bad, neutral, good.