Advanced Machine Learning

Assignment 1: Neural Networks

Osama Bin Zahir

In [113…

```
%%capture
# Installing required packages
!pip install tensorflow
!pip install tensorflow-datasets
```

In [114…

```
(imdb_a_train, imdb_b_train), (imdb_a_test, imdb_b_test) = imdb.load_data(num_words=10
```

In [115…

```
# Importing required Libraries

import numpy as np
from tensorflow.keras.datasets import imdb
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
```

In [116…

```
max([max(sequence) for sequence in imdb_a_train])
```

Out[116]:

```
9999
```

In [117…

```
# Preparing the data for the model

# Retrieving a dictionary mapping words to their index in the IMDB dataset
word_index = imdb.get_word_index()

# Creating a reverse dictionary and mapping the indices back to the original words
reverse_word_index = dict(
    [(value, key) for (key, value) in word_index.items()])
# Create models with different configurations
# Decoding reviews from the IMDB dataset
decoded_review = " ".join([reverse_word_index.get(i - 3, "?") for i in imdb_a_train[0]
```

In [118…

```
# Encoding the integer sequences via multi-hot encoding

import numpy as np
def vectorize_sequences(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        for j in sequence:
            results[i, j] = 1.
    return results
imdb_train = vectorize_sequences(imdb_a_train)
imdb_test = vectorize_sequences(imdb_a_test)
```

In [119…

```
imdb_train[0]
```

Out[119]: array([0., 1., 1., ..., 0., 0., 0.])

In [120…
```
value_train = np.asarray(imdb_b_train).astype("float32")
value_test = np.asarray(imdb_b_test).astype("float32")
```

In [121…
```
# Create models with different configurations
# One hidden layer and 32 hidden units using Tanh activation function instead of relu

model_one_hidden_layer = Sequential()
model_one_hidden_layer.add(Dense(32, activation='tanh'))
model_one_hidden_layer.add(Dense(1, activation='sigmoid'))
```

In [122…
```
# Compiling model using MSE

model_one_hidden_layer.compile(optimizer='adam', loss='mean_squared_error', metrics=['
```

In [123…
```
# Validation

a_val = imdb_train[:10000]
partial_a_train = imdb_train[10000:]
b_val = value_train[:10000]
partial_b_train = value_train[10000:]
```

In [124…
```
# Training the model with validation set

history_one_hidden_layer = model_one_hidden_layer.fit(partial_a_train,
                                                      partial_b_train,
                                                      epochs=20,
                                                      batch_size=512,
                                                      validation_data=(a_val,b_val))
```

```
Epoch 1/20
30/30 [==============================] - 3s 43ms/step - loss: 0.1534 - accuracy: 0.80
93 - val_loss: 0.1060 - val_accuracy: 0.8742
Epoch 2/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0769 - accuracy: 0.91
63 - val_loss: 0.0891 - val_accuracy: 0.8898
Epoch 3/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0560 - accuracy: 0.94
23 - val_loss: 0.0854 - val_accuracy: 0.8893
Epoch 4/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0436 - accuracy: 0.96
01 - val_loss: 0.0841 - val_accuracy: 0.8869
Epoch 5/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0349 - accuracy: 0.97
07 - val_loss: 0.0855 - val_accuracy: 0.8848
Epoch 6/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0283 - accuracy: 0.97
82 - val_loss: 0.0863 - val_accuracy: 0.8811
Epoch 7/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0236 - accuracy: 0.98
33 - val_loss: 0.0876 - val_accuracy: 0.8795
Epoch 8/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0199 - accuracy: 0.98
77 - val_loss: 0.0896 - val_accuracy: 0.8790
Epoch 9/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0168 - accuracy: 0.98
99 - val_loss: 0.0915 - val_accuracy: 0.8756
Epoch 10/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0140 - accuracy: 0.99
24 - val_loss: 0.0928 - val_accuracy: 0.8764
Epoch 11/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0120 - accuracy: 0.99
38 - val_loss: 0.0943 - val_accuracy: 0.8737
Epoch 12/20
30/30 [==============================] - 0s 17ms/step - loss: 0.0103 - accuracy: 0.99
45 - val_loss: 0.0960 - val_accuracy: 0.8721
Epoch 13/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0089 - accuracy: 0.99
53 - val_loss: 0.0973 - val_accuracy: 0.8716
Epoch 14/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0079 - accuracy: 0.99
59 - val_loss: 0.0988 - val_accuracy: 0.8705
Epoch 15/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0071 - accuracy: 0.99
63 - val_loss: 0.1004 - val_accuracy: 0.8680
Epoch 16/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0064 - accuracy: 0.99
65 - val_loss: 0.1007 - val_accuracy: 0.8689
Epoch 17/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0058 - accuracy: 0.99
67 - val_loss: 0.1018 - val_accuracy: 0.8678
Epoch 18/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0053 - accuracy: 0.99
69 - val_loss: 0.1028 - val_accuracy: 0.8665
Epoch 19/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0049 - accuracy: 0.99
70 - val_loss: 0.1039 - val_accuracy: 0.8654
Epoch 20/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0046 - accuracy: 0.99
71 - val_loss: 0.1044 - val_accuracy: 0.8660
```

In [125…

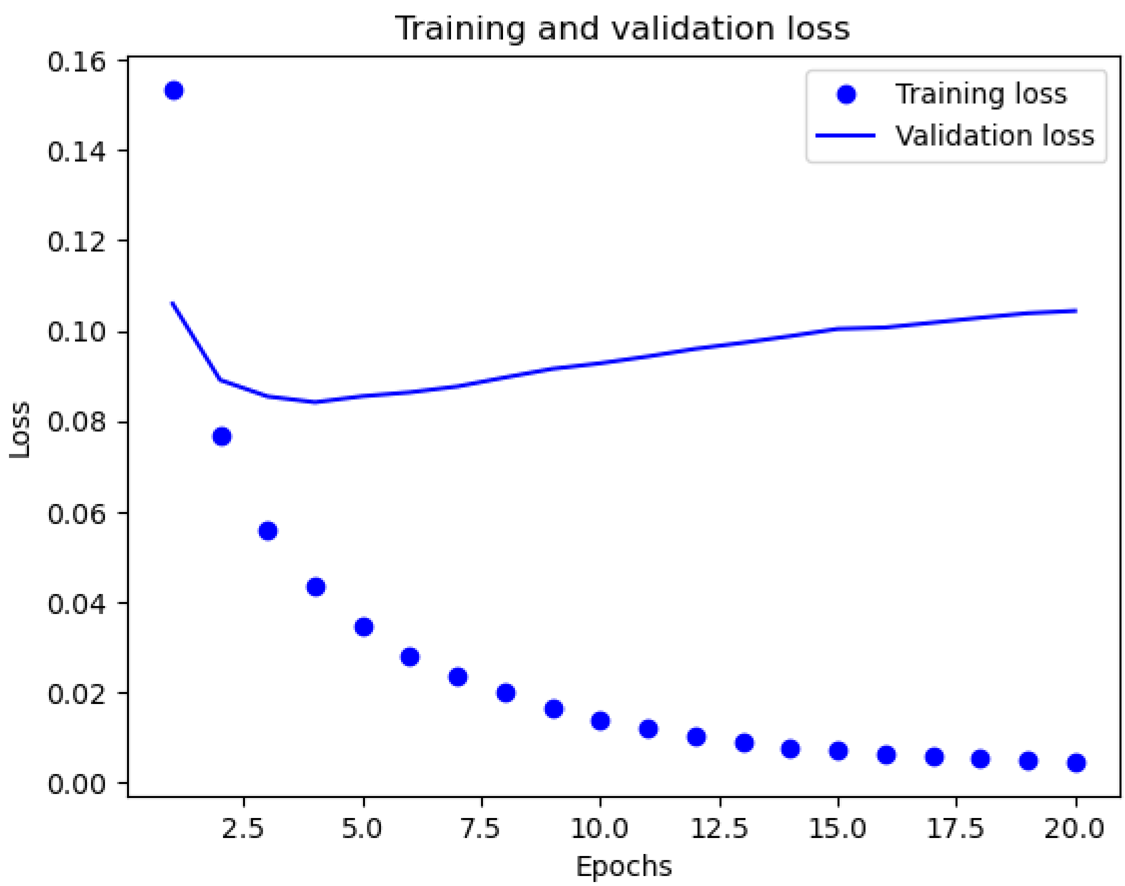
```
history_dict = history_one_hidden_layer.history
history_dict.keys()
```

Out[125]:

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [126…

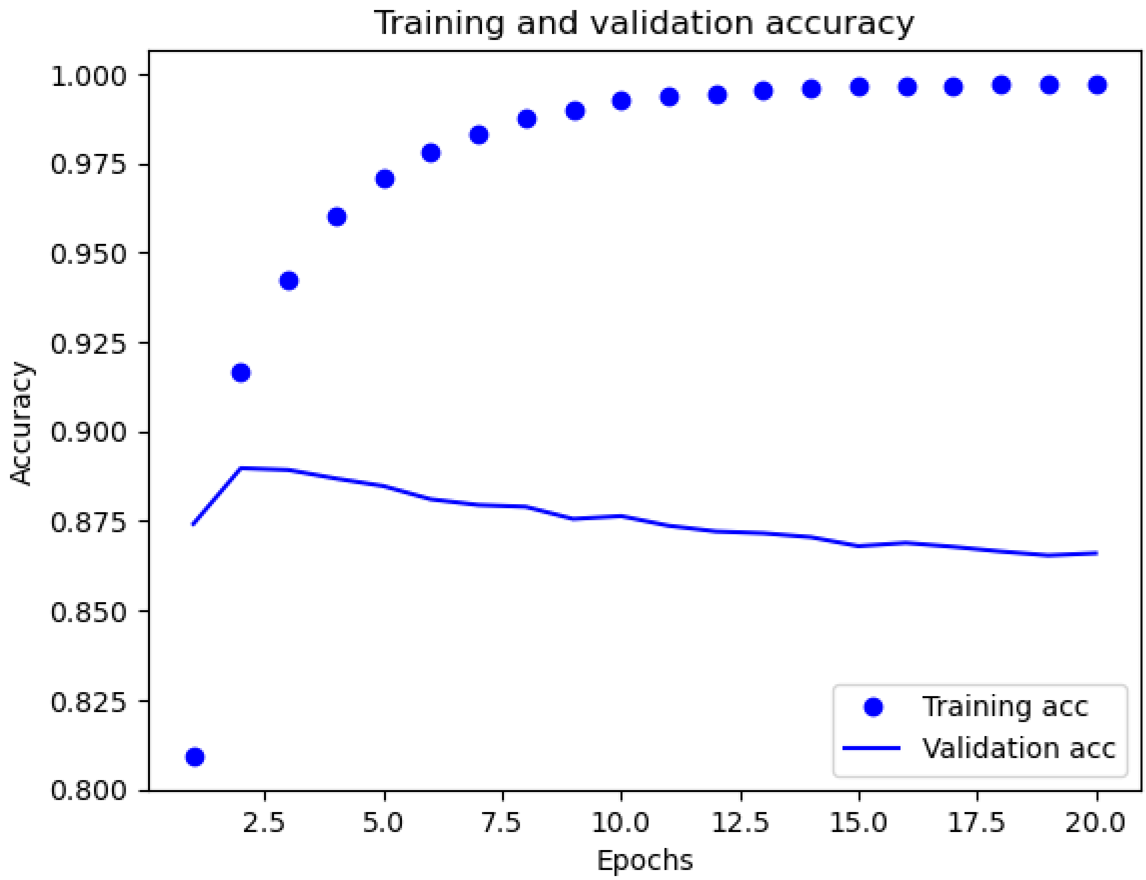
```
# Plotting the training and validation loss

import matplotlib.pyplot as plt
history_dict = history_one_hidden_layer.history
loss_values = history_dict["loss"]
val_loss_values = history_dict["val_loss"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, "bo", label="Training loss")
plt.plot(epochs, val_loss_values, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```



In [127…

```
# Plotting the training and validation accuracy

plt.clf()
acc = history_dict["accuracy"]
val_acc = history_dict["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
```

```
plt.legend()
plt.show()
```



In [128…

```
results = model_one_hidden_layer.evaluate(imdb_test,value_test)
```

```
782/782 [==============================] - 2s 2ms/step - loss: 0.1132 - accuracy: 0.8
554
```

In [129…

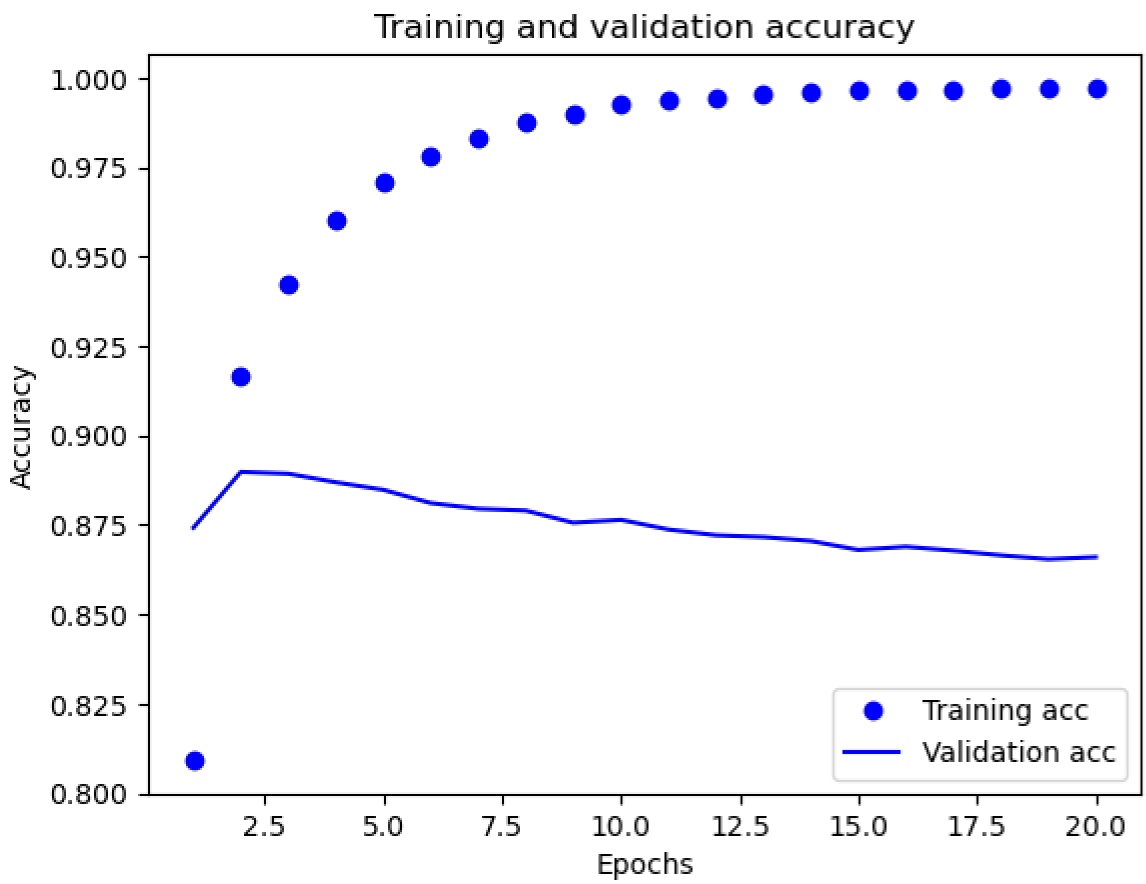
```
# Adding Dropout Layer & Regulaizers

from tensorflow import keras
from tensorflow.keras import layers
from keras.layers import Dense
from tensorflow.keras import regularizers
from keras.layers import Dropout


model_one_hidden_layer = Sequential()
model_one_hidden_layer.add(Dense(32, activation='tanh', activity_regularizer=regulariz
model_one_hidden_layer.add(Dropout(0.5))
model_one_hidden_layer.add(Dense(1, activation='sigmoid'))

# compiling the model

model_one_hidden_layer.compile(optimizer="adam",
                              loss="mean_squared_error",
                              metrics=["accuracy"])

# Validating the model

a_val = imdb_train[:10000]
```

```
partial_a_train = imdb_train[10000:]
b_val = value_train[:10000]
partial_b_train = value_train[10000:]

history = model_one_hidden_layer.fit(partial_a_train,
                                     partial_b_train,
                                     epochs=20,
                                     batch_size=512,
                                     validation_data=(a_val,b_val))
```

```
Epoch 1/20
30/30 [==============================] - 2s 47ms/step - loss: 0.1853 - accuracy: 0.77
97 - val_loss: 0.1419 - val_accuracy: 0.8633
Epoch 2/20
30/30 [==============================] - 0s 15ms/step - loss: 0.1202 - accuracy: 0.90
12 - val_loss: 0.1214 - val_accuracy: 0.8882
Epoch 3/20
30/30 [==============================] - 0s 17ms/step - loss: 0.0999 - accuracy: 0.93
16 - val_loss: 0.1153 - val_accuracy: 0.8889
Epoch 4/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0866 - accuracy: 0.94
89 - val_loss: 0.1128 - val_accuracy: 0.8869
Epoch 5/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0772 - accuracy: 0.96
01 - val_loss: 0.1124 - val_accuracy: 0.8825
Epoch 6/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0699 - accuracy: 0.96
83 - val_loss: 0.1137 - val_accuracy: 0.8772
Epoch 7/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0644 - accuracy: 0.97
52 - val_loss: 0.1156 - val_accuracy: 0.8739
Epoch 8/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0596 - accuracy: 0.97
87 - val_loss: 0.1163 - val_accuracy: 0.8737
Epoch 9/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0553 - accuracy: 0.98
31 - val_loss: 0.1187 - val_accuracy: 0.8682
Epoch 10/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0515 - accuracy: 0.98
67 - val_loss: 0.1202 - val_accuracy: 0.8657
Epoch 11/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0485 - accuracy: 0.98
88 - val_loss: 0.1224 - val_accuracy: 0.8622
Epoch 12/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0460 - accuracy: 0.99
01 - val_loss: 0.1251 - val_accuracy: 0.8561
Epoch 13/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0437 - accuracy: 0.99
16 - val_loss: 0.1276 - val_accuracy: 0.8521
Epoch 14/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0410 - accuracy: 0.99
35 - val_loss: 0.1294 - val_accuracy: 0.8499
Epoch 15/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0393 - accuracy: 0.99
49 - val_loss: 0.1322 - val_accuracy: 0.8473
Epoch 16/20
30/30 [==============================] - 1s 29ms/step - loss: 0.0375 - accuracy: 0.99
55 - val_loss: 0.1342 - val_accuracy: 0.8447
Epoch 17/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0360 - accuracy: 0.99
60 - val_loss: 0.1366 - val_accuracy: 0.8399
Epoch 18/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0350 - accuracy: 0.99
67 - val_loss: 0.1389 - val_accuracy: 0.8369
Epoch 19/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0335 - accuracy: 0.99
67 - val_loss: 0.1420 - val_accuracy: 0.8324
Epoch 20/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0328 - accuracy: 0.99
67 - val_loss: 0.1425 - val_accuracy: 0.8329
```

In [130…

```
plt.clf()
acc = history_dict["accuracy"]
val_acc = history_dict["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



In [131…

```
results = model_one_hidden_layer.evaluate(imdb_test,value_test)
```

```
782/782 [==============================] - 3s 3ms/step - loss: 0.1526 - accuracy: 0.8
188
```

In [132…

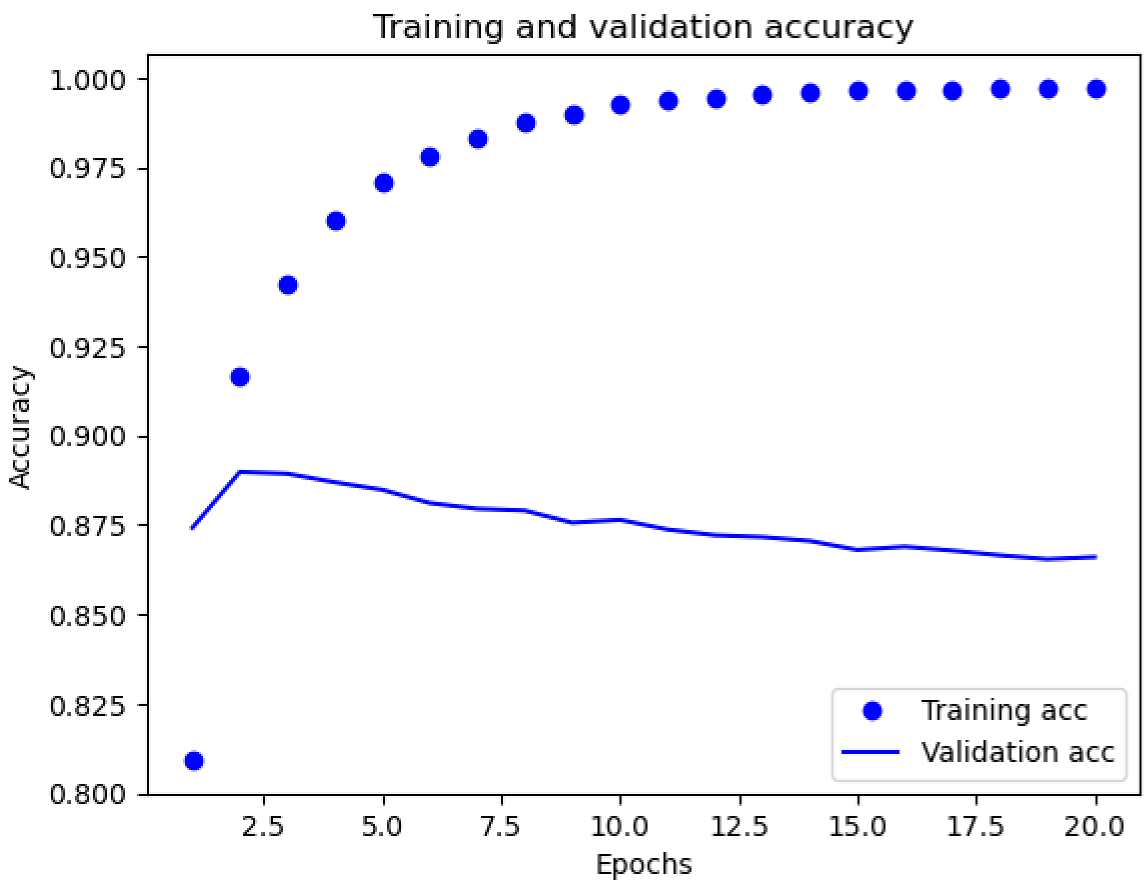
```
# Adding more hidden layers to examine how it affects the accuracy. Here three hidden

from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras import regularizers

model_three_hidden_layers = Sequential()
model_three_hidden_layers.add(Dense(32, activation='tanh', activity_regularizer=regula
model_three_hidden_layers.add(Dropout(0.5))
model_three_hidden_layers.add(Dense(32, activation='tanh', activity_regularizer=regula
model_three_hidden_layers.add(Dropout(0.5))
model_three_hidden_layers.add(Dense(32, activation='tanh', activity_regularizer=regula
model_three_hidden_layers.add(Dropout(0.5))
model_three_hidden_layers.add(Dense(1, activation='sigmoid'))  # Output layer
```

```
# Compliling the model

model_three_hidden_layers.compile(optimizer="adam",
                                loss="mean_squared_error",
                                metrics=["accuracy"])
# Data Splitting

a_val = imdb_train[:10000]
partial_a_train = imdb_train[10000:]
b_val = value_train[:10000]
partial_b_train = value_train[10000:]

history = model_three_hidden_layers.fit(partial_a_train,
                                     partial_b_train,
                                     epochs=20,
                                     batch_size=512,
                                     validation_data=(a_val,b_val))
```

```
Epoch 1/20
30/30 [==============================] - 5s 85ms/step - loss: 0.2446 - accuracy: 0.70
59 - val_loss: 0.1605 - val_accuracy: 0.8634
Epoch 2/20
30/30 [==============================] - 1s 23ms/step - loss: 0.1744 - accuracy: 0.87
83 - val_loss: 0.1401 - val_accuracy: 0.8788
Epoch 3/20
30/30 [==============================] - 1s 31ms/step - loss: 0.1466 - accuracy: 0.90
87 - val_loss: 0.1371 - val_accuracy: 0.8754
Epoch 4/20
30/30 [==============================] - 1s 27ms/step - loss: 0.1294 - accuracy: 0.92
55 - val_loss: 0.1372 - val_accuracy: 0.8711
Epoch 5/20
30/30 [==============================] - 1s 32ms/step - loss: 0.1162 - accuracy: 0.93
97 - val_loss: 0.1398 - val_accuracy: 0.8646
Epoch 6/20
30/30 [==============================] - 1s 27ms/step - loss: 0.1084 - accuracy: 0.94
57 - val_loss: 0.1395 - val_accuracy: 0.8621
Epoch 7/20
30/30 [==============================] - 1s 29ms/step - loss: 0.1013 - accuracy: 0.95
06 - val_loss: 0.1398 - val_accuracy: 0.8581
Epoch 8/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0947 - accuracy: 0.95
65 - val_loss: 0.1409 - val_accuracy: 0.8594
Epoch 9/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0892 - accuracy: 0.96
07 - val_loss: 0.1420 - val_accuracy: 0.8531
Epoch 10/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0851 - accuracy: 0.96
33 - val_loss: 0.1426 - val_accuracy: 0.8509
Epoch 11/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0806 - accuracy: 0.96
65 - val_loss: 0.1464 - val_accuracy: 0.8454
Epoch 12/20
30/30 [==============================] - 1s 27ms/step - loss: 0.0747 - accuracy: 0.97
29 - val_loss: 0.1483 - val_accuracy: 0.8422
Epoch 13/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0721 - accuracy: 0.97
37 - val_loss: 0.1467 - val_accuracy: 0.8440
Epoch 14/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0687 - accuracy: 0.97
66 - val_loss: 0.1482 - val_accuracy: 0.8407
Epoch 15/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0655 - accuracy: 0.97
72 - val_loss: 0.1498 - val_accuracy: 0.8381
Epoch 16/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0618 - accuracy: 0.97
99 - val_loss: 0.1503 - val_accuracy: 0.8377
Epoch 17/20
30/30 [==============================] - 1s 27ms/step - loss: 0.0602 - accuracy: 0.98
15 - val_loss: 0.1518 - val_accuracy: 0.8332
Epoch 18/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0586 - accuracy: 0.98
28 - val_loss: 0.1530 - val_accuracy: 0.8332
Epoch 19/20
30/30 [==============================] - 1s 39ms/step - loss: 0.0564 - accuracy: 0.98
31 - val_loss: 0.1546 - val_accuracy: 0.8298
Epoch 20/20
30/30 [==============================] - 1s 43ms/step - loss: 0.0541 - accuracy: 0.98
58 - val_loss: 0.1539 - val_accuracy: 0.8304
```

In [133…

```
plt.clf()
acc = history_dict["accuracy"]
val_acc = history_dict["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



In [134…

```
results = model_three_hidden_layers.evaluate(imdb_test,value_test)
```

```
782/782 [==============================] - 2s 3ms/step - loss: 0.1644 - accuracy: 0.8
114
```

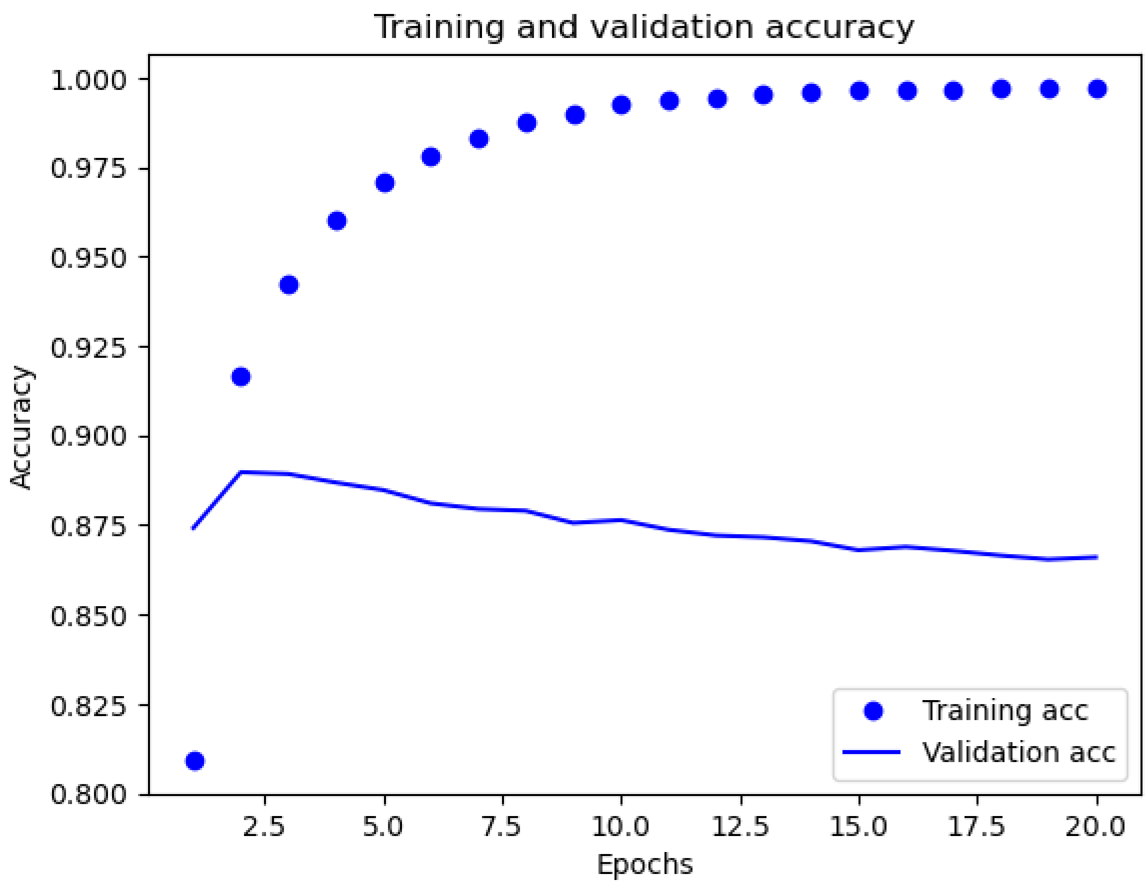
In [135…

```
# Building a model with 64 hidden units and one hidden layer to examine the accuracy

from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras import regularizers

model_one_hidden_layer_64_units = Sequential()
model_one_hidden_layer_64_units.add(Dense(64, activation='tanh', activity_regularizer=
model_one_hidden_layer_64_units.add(Dropout(0.5))
model_one_hidden_layer_64_units.add(Dense(1, activation='sigmoid'))  # Output layer

# Compliling the model

model_one_hidden_layer_64_units.compile(optimizer="adam",
                                        loss="mean_squared_error",
```

```
                                 metrics=["accuracy"])
# Data Splitting

a_val = imdb_train[:10000]
partial_a_train = imdb_train[10000:]
b_val = value_train[:10000]
partial_b_train = value_train[10000:]

history = model_one_hidden_layer_64_units.fit(partial_a_train,
                                     partial_b_train,
                                     epochs=20,
                                     batch_size=512,
                                     validation_data=(a_val,b_val))
```

```
Epoch 1/20
30/30 [==============================] - 3s 53ms/step - loss: 0.1677 - accuracy: 0.80
76 - val_loss: 0.1254 - val_accuracy: 0.8801
Epoch 2/20
30/30 [==============================] - 1s 25ms/step - loss: 0.1033 - accuracy: 0.92
12 - val_loss: 0.1160 - val_accuracy: 0.8900
Epoch 3/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0849 - accuracy: 0.94
48 - val_loss: 0.1147 - val_accuracy: 0.8839
Epoch 4/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0722 - accuracy: 0.96
29 - val_loss: 0.1154 - val_accuracy: 0.8790
Epoch 5/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0639 - accuracy: 0.97
25 - val_loss: 0.1179 - val_accuracy: 0.8742
Epoch 6/20
30/30 [==============================] - 1s 44ms/step - loss: 0.0576 - accuracy: 0.97
81 - val_loss: 0.1206 - val_accuracy: 0.8689
Epoch 7/20
30/30 [==============================] - 1s 45ms/step - loss: 0.0520 - accuracy: 0.98
39 - val_loss: 0.1238 - val_accuracy: 0.8636
Epoch 8/20
30/30 [==============================] - 1s 39ms/step - loss: 0.0480 - accuracy: 0.98
79 - val_loss: 0.1268 - val_accuracy: 0.8584
Epoch 9/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0448 - accuracy: 0.98
89 - val_loss: 0.1294 - val_accuracy: 0.8553
Epoch 10/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0422 - accuracy: 0.99
15 - val_loss: 0.1327 - val_accuracy: 0.8488
Epoch 11/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0394 - accuracy: 0.99
31 - val_loss: 0.1383 - val_accuracy: 0.8404
Epoch 12/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0374 - accuracy: 0.99
43 - val_loss: 0.1413 - val_accuracy: 0.8349
Epoch 13/20
30/30 [==============================] - 1s 36ms/step - loss: 0.0360 - accuracy: 0.99
41 - val_loss: 0.1431 - val_accuracy: 0.8359
Epoch 14/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0344 - accuracy: 0.99
52 - val_loss: 0.1465 - val_accuracy: 0.8326
Epoch 15/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0329 - accuracy: 0.99
65 - val_loss: 0.1482 - val_accuracy: 0.8293
Epoch 16/20
30/30 [==============================] - 1s 36ms/step - loss: 0.0312 - accuracy: 0.99
71 - val_loss: 0.1515 - val_accuracy: 0.8268
Epoch 17/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0302 - accuracy: 0.99
71 - val_loss: 0.1552 - val_accuracy: 0.8240
Epoch 18/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0294 - accuracy: 0.99
67 - val_loss: 0.1570 - val_accuracy: 0.8221
Epoch 19/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0284 - accuracy: 0.99
73 - val_loss: 0.1595 - val_accuracy: 0.8186
Epoch 20/20
30/30 [==============================] - 1s 43ms/step - loss: 0.0278 - accuracy: 0.99
74 - val_loss: 0.1610 - val_accuracy: 0.8176
```

In [136…
```
plt.clf()
acc = history_dict["accuracy"]
val_acc = history_dict["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



In [137…
```
results =  model_one_hidden_layer_64_units.evaluate(imdb_test,value_test)
```

```
782/782 [==============================] - 3s 3ms/step - loss: 0.1711 - accuracy: 0.8
002
```

- Neural Network Configurations for IMDB Sentiment Analysis
- Introduction

This report presents the results of training several neural network models with different configurations for sentiment analysis on the IMDB dataset. The goal is to determine the optimal model architecture for accurately classifying movie reviews as positive or negative.

- Experimental Setup

Dataset: The IMDB dataset consists of 50,000 movie reviews labeled as positive or negative. Model Configurations: Four different neural network configurations were evaluated: One hidden layer with 32 units and Tanh activation function One hidden layer with 32 units, Tanh activation function, and dropout regularization Three hidden layers with 32 units each, Tanh activation function, and dropout regularization One hidden layer with 64 units, Tanh activation function, and dropout regularization

- Results and Analysis

- Model 1: One Hidden Layer (32 Units, Tanh Activation) Accuracy: 85.54% Validation Accuracy: 86.60% Observations: The model shows a good performance with decent accuracy on both training and validation sets. However, there is a slight overfitting as the training accuracy is slightly higher than the validation accuracy.

- Model 2: One Hidden Layer (32 Units, Tanh Activation, Dropout Regularization) Accuracy: 81.88% Validation Accuracy: 83.29% Observations: Introducing dropout regularization slightly reduces overfitting compared to Model 1, but there is still room for improvement in validation accuracy.

- Model 3: Three Hidden Layers (32 Units, Tanh Activation, Dropout Regularization) Accuracy: 81.14% Validation Accuracy: 83.04% Observations: Adding more hidden layers does not significantly improve performance. The model seems to suffer from overfitting, similar to Model 2.

- Model 4: One Hidden Layer (64 Units, Tanh Activation, Dropout Regularization) Accuracy: 80.02% Validation Accuracy: 81.76% Observations: Increasing the number of units in the hidden layer does not lead to better performance. The model exhibits similar overfitting issues as previous configurations.

Model	Validation Accuracy	Test Loss	Test Accuracy
One Hidden Layer (32 units)	0.8660	0.1132	0.8554
One Hidden Layer (32 units)	0.8329	0.1526	0.8188
Three Hidden Layers (32 units)	0.8304	0.1644	0.8114
One Hidden Layer (64 units)	0.8176	0.1711	0.8002

- Conclusion

Based on the experiments conducted, the model with one hidden layer consisting of 32 units and Tanh activation function (Model 1) performs the best, achieving the highest accuracy on both the training and validation sets. Introducing dropout regularization helps mitigate overfitting to some extent, but adding more hidden layers or increasing the number of units does not yield significant improvements. Therefore, Model 1 is recommended as the optimal configuration for sentiment analysis on the IMDB dataset. However, further experimentation with hyperparameter tuning and different architectures could potentially lead to even better results.