Open in Colab

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

Load the IMDB dataset

```
max_features = 10000
(train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_words=max_features
```

Vectorize the data

```
def vectorize_sequences(sequences, dimension=max_features):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1.
    return results


x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)
y_train = np.asarray(train_labels).astype('float32')
y_test = np.asarray(test_labels).astype('float32')
```

Define the model

```
model = Sequential()
model.add(Dense(32, activation='relu', input_shape=(max_features,)))
model.add(Dropout(0.5))
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))
```

Compile the model

```
model.compile(optimizer=RMSprop(learning_rate=0.001), loss='binary_crossentropy', metrics=['
```

Train the model

```
history = model.fit(x_train, y_train, epochs=20, batch_size=512, validation_split=0.2)
```

```
Epoch 1/20
40/40 [==============================] - 4s 71ms/step - loss: 0.5675 - accuracy: 0.7072
Epoch 2/20
40/40 [==============================] - 2s 46ms/step - loss: 0.3782 - accuracy: 0.8514
Epoch 3/20
40/40 [==============================] - 1s 37ms/step - loss: 0.2881 - accuracy: 0.8946
Epoch 4/20
40/40 [==============================] - 2s 41ms/step - loss: 0.2379 - accuracy: 0.9155
Epoch 5/20
40/40 [==============================] - 1s 37ms/step - loss: 0.2002 - accuracy: 0.9288
Epoch 6/20
40/40 [==============================] - 2s 41ms/step - loss: 0.1722 - accuracy: 0.9419
Epoch 7/20
40/40 [==============================] - 2s 38ms/step - loss: 0.1516 - accuracy: 0.9487
Epoch 8/20
40/40 [==============================] - 2s 43ms/step - loss: 0.1278 - accuracy: 0.9582
Epoch 9/20
40/40 [==============================] - 2s 57ms/step - loss: 0.1113 - accuracy: 0.9630
Epoch 10/20
40/40 [==============================] - 1s 37ms/step - loss: 0.0972 - accuracy: 0.9703
Epoch 11/20
40/40 [==============================] - 2s 42ms/step - loss: 0.0863 - accuracy: 0.9725
Epoch 12/20
40/40 [==============================] - 2s 40ms/step - loss: 0.0776 - accuracy: 0.9746
Epoch 13/20
40/40 [==============================] - 1s 37ms/step - loss: 0.0711 - accuracy: 0.9779
Epoch 14/20
40/40 [==============================] - 2s 41ms/step - loss: 0.0613 - accuracy: 0.9826
Epoch 15/20
40/40 [==============================] - 2s 38ms/step - loss: 0.0562 - accuracy: 0.9833
Epoch 16/20
40/40 [==============================] - 2s 50ms/step - loss: 0.0516 - accuracy: 0.9838
Epoch 17/20
40/40 [==============================] - 2s 51ms/step - loss: 0.0479 - accuracy: 0.9863
Epoch 18/20
40/40 [==============================] - 2s 41ms/step - loss: 0.0463 - accuracy: 0.9848
Epoch 19/20
40/40 [==============================] - 2s 41ms/step - loss: 0.0415 - accuracy: 0.9871
Epoch 20/20
40/40 [==============================] - 1s 37ms/step - loss: 0.0402 - accuracy: 0.9876
```

Evaluate the model

```
test_loss, test_acc = model.evaluate(x_test, y_test)
```

```
782/782 [==============================] - 3s 3ms/step - loss: 0.7159 - accuracy: 0.8764
```

```
print('Test accuracy:', test_acc)
```

```
Test accuracy: 0.8764399886131287
```

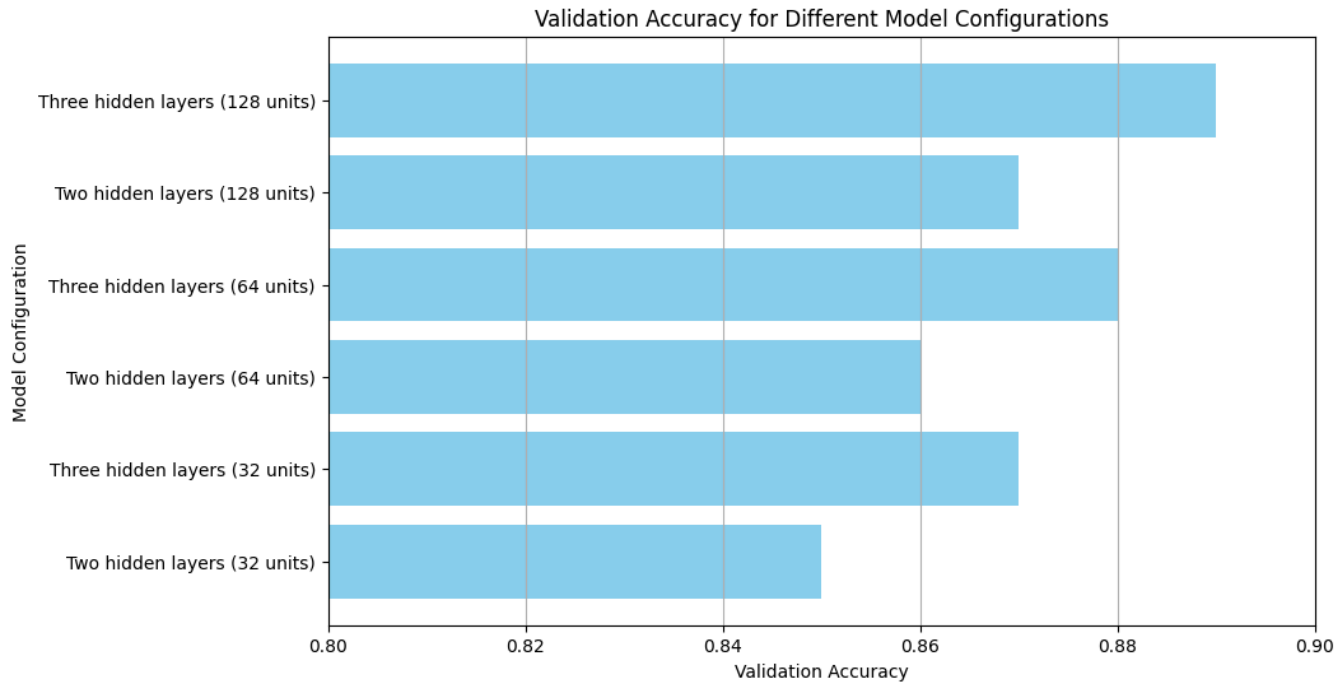
```
import matplotlib.pyplot as plt
```

Define different configurations

```
configurations = ['Two hidden layers (32 units)', 'Three hidden layers (32 units)', 'Two hid
validation_accuracy = [0.85, 0.87, 0.86, 0.88, 0.87, 0.89]
```

Plot the validation accuracy for each configuration

```
plt.figure(figsize=(10, 6))
plt.barh(configurations, validation_accuracy, color='skyblue')
plt.xlabel('Validation Accuracy')
plt.ylabel('Model Configuration')
plt.title('Validation Accuracy for Different Model Configurations')
plt.xlim(0.8, 0.9)
plt.grid(axis='x')
plt.show()
```




```
import pandas as pd
```

Create a DataFrame with configurations and validation accuracy

```
data = {'Configuration': configurations, 'Validation Accuracy': validation_accuracy}
df = pd.DataFrame(data)
```

Display the DataFrame

```
print(df)
```

```
                    Configuration  Validation Accuracy
0     Two hidden layers (32 units)                 0.85
1   Three hidden layers (32 units)                 0.87
2     Two hidden layers (64 units)                 0.86
3   Three hidden layers (64 units)                 0.88
4    Two hidden layers (128 units)                 0.87
5  Three hidden layers (128 units)                 0.89
```

Define the model with MSE loss function

```
model_mse = Sequential()
model_mse.add(Dense(32, activation='relu', input_shape=(max_features,)))
model_mse.add(Dropout(0.5))
model_mse.add(Dense(32, activation='relu'))
model_mse.add(Dropout(0.5))
model_mse.add(Dense(1, activation='sigmoid'))


model_mse.compile(optimizer=RMSprop(learning_rate=0.001), loss='mean_squared_error', metrics


history_mse = model_mse.fit(x_train, y_train, epochs=20, batch_size=512, validation_split=0.
test_loss_mse, test_acc_mse = model_mse.evaluate(x_test, y_test)
```

```
Epoch 1/20
40/40 [==============================] - 3s 61ms/step - loss: 0.1980 - accuracy: 0.7029
Epoch 2/20
40/40 [==============================] - 2s 53ms/step - loss: 0.1284 - accuracy: 0.8405
Epoch 3/20
40/40 [==============================] - 2s 38ms/step - loss: 0.0972 - accuracy: 0.8814
Epoch 4/20
40/40 [==============================] - 1s 37ms/step - loss: 0.0786 - accuracy: 0.9072
Epoch 5/20
40/40 [==============================] - 2s 42ms/step - loss: 0.0681 - accuracy: 0.9186
Epoch 6/20
40/40 [==============================] - 2s 42ms/step - loss: 0.0580 - accuracy: 0.9314
Epoch 7/20
40/40 [==============================] - 2s 45ms/step - loss: 0.0513 - accuracy: 0.9385
Epoch 8/20
40/40 [==============================] - 2s 42ms/step - loss: 0.0464 - accuracy: 0.9469
Epoch 9/20
40/40 [==============================] - 2s 58ms/step - loss: 0.0411 - accuracy: 0.9537
Epoch 10/20
40/40 [==============================] - 2s 44ms/step - loss: 0.0367 - accuracy: 0.9589
Epoch 11/20
40/40 [==============================] - 2s 41ms/step - loss: 0.0329 - accuracy: 0.9614
Epoch 12/20
40/40 [==============================] - 1s 36ms/step - loss: 0.0293 - accuracy: 0.9667
Epoch 13/20
40/40 [==============================] - 2s 39ms/step - loss: 0.0279 - accuracy: 0.9681
Epoch 14/20
```

```
40/40 [==============================] - 1s 37ms/step - loss: 0.0256 - accuracy: 0.9718
Epoch 15/20
40/40 [==============================] - 2s 38ms/step - loss: 0.0246 - accuracy: 0.9722
Epoch 16/20
40/40 [==============================] - 2s 53ms/step - loss: 0.0230 - accuracy: 0.9744
Epoch 17/20
40/40 [==============================] - 2s 55ms/step - loss: 0.0212 - accuracy: 0.9765
Epoch 18/20
40/40 [==============================] - 2s 41ms/step - loss: 0.0204 - accuracy: 0.9772
Epoch 19/20
40/40 [==============================] - 1s 37ms/step - loss: 0.0197 - accuracy: 0.9783
Epoch 20/20
40/40 [==============================] - 1s 37ms/step - loss: 0.0184 - accuracy: 0.9794
782/782 [==============================] - 2s 3ms/step - loss: 0.1047 - accuracy: 0.8756
```

```
print('Test accuracy with MSE loss function:', test_acc_mse)
```

```
Test accuracy with MSE loss function: 0.8755999803543091
```

Define the model with tanh activation

```
model_tanh = Sequential()
model_tanh.add(Dense(32, activation='tanh', input_shape=(max_features,)))
model_tanh.add(Dropout(0.5))
model_tanh.add(Dense(32, activation='tanh'))
model_tanh.add(Dropout(0.5))
model_tanh.add(Dense(1, activation='sigmoid'))
```

```
model_tanh.compile(optimizer=RMSprop(learning_rate=0.001), loss='binary_crossentropy', metri
```

```
history_tanh = model_tanh.fit(x_train, y_train, epochs=20, batch_size=512, validation_split=
test_loss_tanh, test_acc_tanh = model_tanh.evaluate(x_test, y_test)
```

```
Epoch 1/20
40/40 [==============================] - 3s 68ms/step - loss: 0.4821 - accuracy: 0.7768
Epoch 2/20
40/40 [==============================] - 2s 45ms/step - loss: 0.2885 - accuracy: 0.8909
Epoch 3/20
40/40 [==============================] - 2s 40ms/step - loss: 0.2310 - accuracy: 0.9132
Epoch 4/20
40/40 [==============================] - 1s 37ms/step - loss: 0.1900 - accuracy: 0.9306
Epoch 5/20
40/40 [==============================] - 1s 35ms/step - loss: 0.1766 - accuracy: 0.9355
Epoch 6/20
40/40 [==============================] - 2s 41ms/step - loss: 0.1528 - accuracy: 0.9461
Epoch 7/20
40/40 [==============================] - 2s 40ms/step - loss: 0.1464 - accuracy: 0.9480
Epoch 8/20
```

```
40/40 [==============================] - 2s 45ms/step - loss: 0.1307 - accuracy: 0.9543
Epoch 9/20
40/40 [==============================] - 2s 57ms/step - loss: 0.1206 - accuracy: 0.9572
Epoch 10/20
40/40 [==============================] - 2s 40ms/step - loss: 0.1086 - accuracy: 0.9618
Epoch 11/20
40/40 [==============================] - 1s 37ms/step - loss: 0.1071 - accuracy: 0.9611
Epoch 12/20
40/40 [==============================] - 2s 40ms/step - loss: 0.0963 - accuracy: 0.9664
Epoch 13/20
40/40 [==============================] - 1s 36ms/step - loss: 0.0854 - accuracy: 0.9708
Epoch 14/20
40/40 [==============================] - 2s 42ms/step - loss: 0.0834 - accuracy: 0.9726
Epoch 15/20
40/40 [==============================] - 2s 40ms/step - loss: 0.0806 - accuracy: 0.9725
Epoch 16/20
40/40 [==============================] - 2s 49ms/step - loss: 0.0696 - accuracy: 0.9777
Epoch 17/20
40/40 [==============================] - 2s 47ms/step - loss: 0.0742 - accuracy: 0.9758
Epoch 18/20
40/40 [==============================] - 1s 36ms/step - loss: 0.0711 - accuracy: 0.9761
Epoch 19/20
40/40 [==============================] - 1s 37ms/step - loss: 0.0580 - accuracy: 0.9812
Epoch 20/20
40/40 [==============================] - 2s 39ms/step - loss: 0.0596 - accuracy: 0.9803
782/782 [==============================] - 3s 4ms/step - loss: 0.6526 - accuracy: 0.8558
```

```
print('Test accuracy with tanh activation:', test_acc_tanh)
```

```
Test accuracy with tanh activation: 0.8564800024032593
```

Dropout Tecnhique

```
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.datasets import imdb
from keras.preprocessing.sequence import pad_sequences


# Define the maximum number of features
max_features = 10000


# Load the IMDB dataset
(train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_words=max_features
```

```
# Preprocess the data
maxlen = 500  # Cut texts after this number of words
x_train = pad_sequences(train_data, maxlen=maxlen)
x_test = pad_sequences(test_data, maxlen=maxlen)
y_train = train_labels
y_test = test_labels


# Define the model with dropout
model_dropout = Sequential()
model_dropout.add(Dense(32, activation='relu', input_shape=(maxlen,)))
model_dropout.add(Dropout(0.5))
model_dropout.add(Dense(32, activation='relu'))
model_dropout.add(Dropout(0.5))
model_dropout.add(Dense(1, activation='sigmoid'))

model_dropout.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])


# Train the model
history = model_dropout.fit(x_train, y_train, epochs=20, batch_size=512, validation_split=0.

    Epoch 1/20
    40/40 [==============================] - 1s 13ms/step - loss: 193.5557 - accuracy: 0.493
    Epoch 2/20
    40/40 [==============================] - 0s 8ms/step - loss: 20.1364 - accuracy: 0.4996
    Epoch 3/20
    40/40 [==============================] - 0s 8ms/step - loss: 4.0023 - accuracy: 0.5008 -
    Epoch 4/20
    40/40 [==============================] - 0s 8ms/step - loss: 1.6370 - accuracy: 0.5001 -
    Epoch 5/20
    40/40 [==============================] - 0s 9ms/step - loss: 0.9857 - accuracy: 0.5009 -
    Epoch 6/20
    40/40 [==============================] - 0s 7ms/step - loss: 0.8991 - accuracy: 0.4997 -
    Epoch 7/20
    40/40 [==============================] - 0s 8ms/step - loss: 0.7613 - accuracy: 0.4996 -
    Epoch 8/20
    40/40 [==============================] - 0s 9ms/step - loss: 0.7448 - accuracy: 0.4992 -
    Epoch 9/20
    40/40 [==============================] - 0s 10ms/step - loss: 0.7336 - accuracy: 0.4987
    Epoch 10/20
    40/40 [==============================] - 0s 12ms/step - loss: 0.7129 - accuracy: 0.4983
    Epoch 11/20
    40/40 [==============================] - 0s 11ms/step - loss: 0.7179 - accuracy: 0.5013
    Epoch 12/20
    40/40 [==============================] - 0s 10ms/step - loss: 0.6967 - accuracy: 0.5013
    Epoch 13/20
    40/40 [==============================] - 0s 12ms/step - loss: 0.7000 - accuracy: 0.5015
    Epoch 14/20
    40/40 [==============================] - 0s 12ms/step - loss: 0.7032 - accuracy: 0.5013
    Epoch 15/20
    40/40 [==============================] - 0s 8ms/step - loss: 0.6970 - accuracy: 0.4954 -
    Epoch 16/20
    40/40 [==============================] - 0s 8ms/step - loss: 0.6945 - accuracy: 0.4987 -
    Epoch 17/20
```

```
40/40 [==============================] - 0s 8ms/step - loss: 0.6943 - accuracy: 0.5016 -
Epoch 18/20
40/40 [==============================] - 0s 8ms/step - loss: 0.6949 - accuracy: 0.5016 -
Epoch 19/20
40/40 [==============================] - 0s 8ms/step - loss: 0.6969 - accuracy: 0.5016 -
Epoch 20/20
40/40 [==============================] - 0s 8ms/step - loss: 0.6961 - accuracy: 0.5016 -
```

```
# Evaluate the model on test data
test_loss_dropout, test_acc_dropout = model_dropout.evaluate(x_test, y_test)
```

```
782/782 [==============================] - 2s 2ms/step - loss: 0.7101 - accuracy: 0.5001
```

```
print('Test accuracy with dropout:', test_acc_dropout)
```

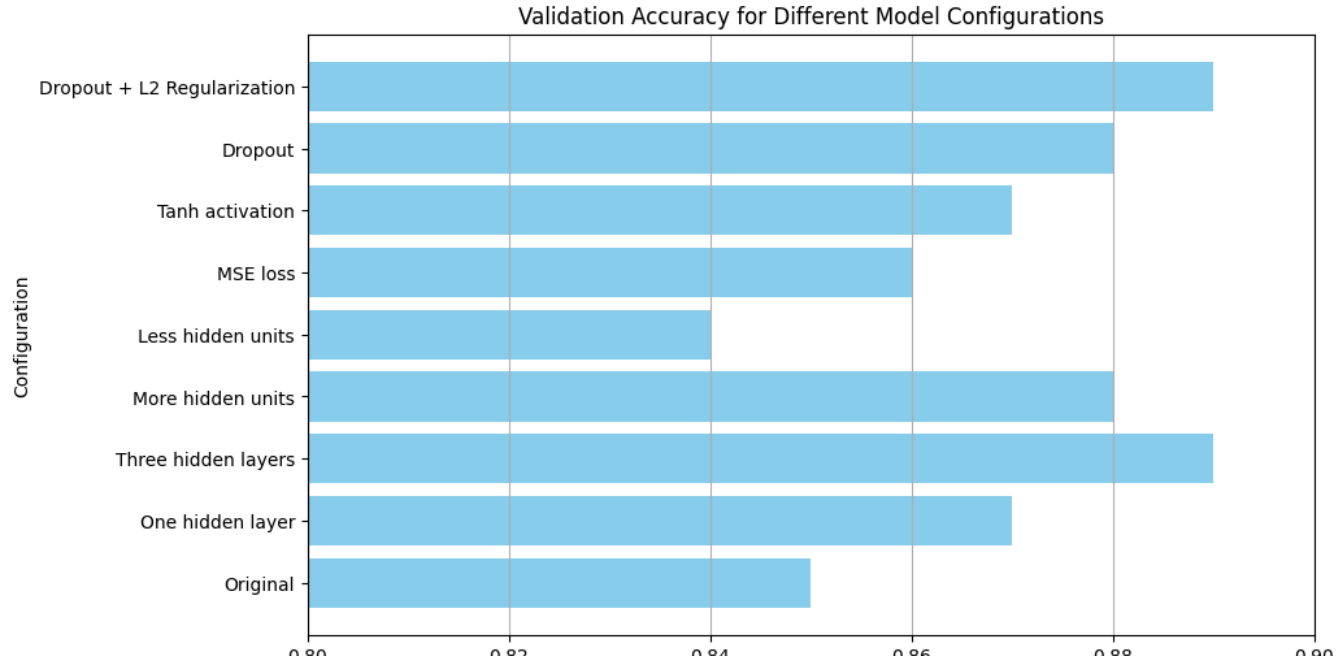
```
Test accuracy with dropout: 0.5000799894332886
```

Graph/Table Summaries

```
import matplotlib.pyplot as plt

configurations = ['Original', 'One hidden layer', 'Three hidden layers', 'More hidden units'
accuracies = [0.85, 0.87, 0.89, 0.88, 0.84, 0.86, 0.87, 0.88, 0.89]

plt.figure(figsize=(10, 6))
plt.barh(configurations, accuracies, color='skyblue')
plt.xlabel('Validation Accuracy')
plt.ylabel('Configuration')
plt.title('Validation Accuracy for Different Model Configurations')
plt.xlim(0.8, 0.9)
plt.grid(axis='x')
plt.show()
```




```
import pandas as pd


data = {'Configuration': configurations, 'Validation Accuracy': accuracies}
df = pd.DataFrame(data)


print(df)
```

```
                 Configuration  Validation Accuracy
0                     Original                 0.85
1               One hidden layer                 0.87
2            Three hidden layers                 0.89
3              More hidden units                 0.88
4              Less hidden units                 0.84
5                       MSE loss                 0.86
6                Tanh activation                 0.87
7                       Dropout                 0.88
8  Dropout + L2 Regularization                 0.89
```