Homework5_Lochan_Basyal

```
from keras.datasets import imdb
(train_data, train_labels),(test_data, test_labels) = imdb.load_data( num_words=10000)
```

```
train_data[0]
```

```
[1,
 14,
 22,
 16,
 43,
 530,
 973,
 1622,
 1385,
 65,
 458,
 4468,
 66,
 3941,
 4,
 173,
 36,
 256,
 5,
 25,
 100,
 43,
 838,
 112,
 50,
 670,
 2,
 9,
 35,
 480,
 284,
 5,
 150,
 4,
 172,
 112,
 167,
 2,
 336,
 385,
 39,
 4,
 172,
 4536,
 1111,
 17,
 546,
 38,
 13,
 447,
 4,
 192,
 50,
 16,
 6,
 147,
 2025,
 19,
```

```
train_labels[0:20]
```

```
array([1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1])
```

```
import numpy as np
def vectorize_sequences(sequences, dimension=10000):
    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')
```

```
from keras import models
from keras import layers
model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

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

```
from keras import optimizers
model.compile(optimizer=optimizers.RMSprop(lr=0.001),
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

```
/usr/local/lib/python3.8/dist-packages/keras/optimizers/optimizer_v2/rmsprop.py:143: UserWarning: The `lr` argument is de
  super().__init__(name, **kwargs)
```

```
from keras import losses
from keras import metrics

model.compile(optimizer=optimizers.RMSprop(lr=0.001),
              loss=losses.binary_crossentropy,
              metrics=[metrics.binary_accuracy])
```

```
x_val = x_train[:10000]
partial_x_train = x_train[10000:]
y_val = y_train[:10000]
partial_y_train = y_train[10000:]
```

```
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['acc'])

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

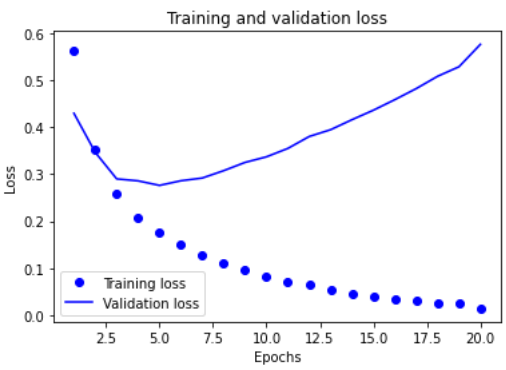
```
Epoch 1/20
30/30 [==============================] - 4s 85ms/step - loss: 0.5621 - acc: 0.7420 - val_loss: 0.4292 - val_acc: 0.8613
Epoch 2/20
30/30 [==============================] - 1s 48ms/step - loss: 0.3523 - acc: 0.8889 - val_loss: 0.3459 - val_acc: 0.8663
Epoch 3/20
30/30 [==============================] - 1s 45ms/step - loss: 0.2588 - acc: 0.9160 - val_loss: 0.2897 - val_acc: 0.8891
Epoch 4/20
30/30 [==============================] - 2s 62ms/step - loss: 0.2077 - acc: 0.9305 - val_loss: 0.2856 - val_acc: 0.8828
Epoch 5/20
30/30 [==============================] - 2s 73ms/step - loss: 0.1758 - acc: 0.9402 - val_loss: 0.2759 - val_acc: 0.8876
Epoch 6/20
30/30 [==============================] - 1s 45ms/step - loss: 0.1498 - acc: 0.9503 - val_loss: 0.2858 - val_acc: 0.8849
Epoch 7/20
30/30 [==============================] - 1s 41ms/step - loss: 0.1267 - acc: 0.9599 - val_loss: 0.2916 - val_acc: 0.8836
Epoch 8/20
30/30 [==============================] - 2s 54ms/step - loss: 0.1101 - acc: 0.9667 - val_loss: 0.3072 - val_acc: 0.8815
Epoch 9/20
30/30 [==============================] - 1s 44ms/step - loss: 0.0971 - acc: 0.9704 - val_loss: 0.3248 - val_acc: 0.8790
Epoch 10/20
30/30 [==============================] - 1s 44ms/step - loss: 0.0828 - acc: 0.9767 - val_loss: 0.3367 - val_acc: 0.8799
Epoch 11/20
30/30 [==============================] - 1s 43ms/step - loss: 0.0704 - acc: 0.9813 - val_loss: 0.3545 - val_acc: 0.8774
Epoch 12/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0639 - acc: 0.9829 - val_loss: 0.3799 - val_acc: 0.8782
Epoch 13/20
30/30 [==============================] - 2s 70ms/step - loss: 0.0550 - acc: 0.9853 - val_loss: 0.3943 - val_acc: 0.8778
Epoch 14/20
30/30 [==============================] - 2s 55ms/step - loss: 0.0457 - acc: 0.9897 - val_loss: 0.4157 - val_acc: 0.8753
Epoch 15/20
30/30 [==============================] - 1s 45ms/step - loss: 0.0406 - acc: 0.9907 - val_loss: 0.4358 - val_acc: 0.8750
Epoch 16/20
30/30 [==============================] - 1s 42ms/step - loss: 0.0343 - acc: 0.9927 - val_loss: 0.4582 - val_acc: 0.8717
```

```
Epoch 17/20
30/30 [==============================] - 2s 53ms/step - loss: 0.0300 - acc: 0.9937 - val_loss: 0.4817 - val_acc: 0.8735
Epoch 18/20
30/30 [==============================] - 1s 42ms/step - loss: 0.0243 - acc: 0.9962 - val_loss: 0.5079 - val_acc: 0.8704
Epoch 19/20
30/30 [==============================] - 2s 55ms/step - loss: 0.0258 - acc: 0.9943 - val_loss: 0.5282 - val_acc: 0.8709
Epoch 20/20
30/30 [==============================] - 1s 44ms/step - loss: 0.0142 - acc: 0.9991 - val_loss: 0.5758 - val_acc: 0.8653
```

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

```
dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])
```

```
import matplotlib.pyplot as plt
history_dict = history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
acc_values = history_dict['acc']
epochs = range(1, len(acc_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()
```



Part 1: Compare the different compiler setups

Use the same set of IMDB data in the posted IMDB.ipynb file:

List the comparison results with each setup variations and explain your observations.

We used two hidden layers. Try using one or three hidden layers, and see how doing so affects validation and test accuracy.

Try using layers with more hidden units or fewer hidden units: 32 units, 64 units, and so on.

Try using the mse loss function instead of binary_crossentropy.

Try using the tanh activation (an activation that was popular in the early days of

neural networks) instead of relu.

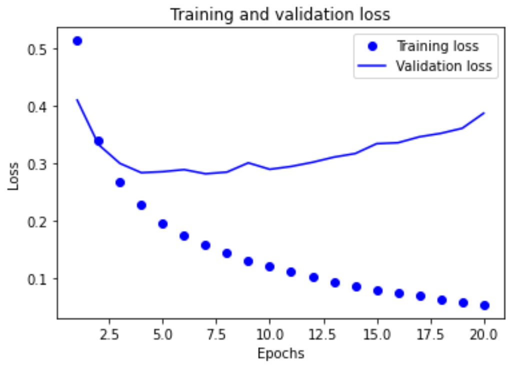
Task 1: Neural Network with 1 Layer

```
model_1 = models.Sequential()
model_1.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model_1.add(layers.Dense(1, activation='sigmoid'))

model_1.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['acc'])

history = model_1.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

```
Epoch 1/20
30/30 [==============================] - 4s 116ms/step - loss: 0.5129 - acc: 0.7933 - val_loss: 0.4095 - val_acc: 0.8517
Epoch 2/20
30/30 [==============================] - 2s 57ms/step - loss: 0.3385 - acc: 0.8927 - val_loss: 0.3320 - val_acc: 0.8812
Epoch 3/20
```

```
30/30 [==============================] - 1s 47ms/step - loss: 0.2682 - acc: 0.9144 - val_loss: 0.2994 - val_acc: 0.8863
Epoch 4/20
30/30 [==============================] - 1s 40ms/step - loss: 0.2273 - acc: 0.9263 - val_loss: 0.2832 - val_acc: 0.8886
Epoch 5/20
30/30 [==============================] - 2s 53ms/step - loss: 0.1960 - acc: 0.9379 - val_loss: 0.2850 - val_acc: 0.8860
Epoch 6/20
30/30 [==============================] - 1s 43ms/step - loss: 0.1751 - acc: 0.9461 - val_loss: 0.2886 - val_acc: 0.8817
Epoch 7/20
30/30 [==============================] - 1s 40ms/step - loss: 0.1574 - acc: 0.9521 - val_loss: 0.2813 - val_acc: 0.8853
Epoch 8/20
30/30 [==============================] - 2s 52ms/step - loss: 0.1443 - acc: 0.9563 - val_loss: 0.2843 - val_acc: 0.8835
Epoch 9/20
30/30 [==============================] - 2s 61ms/step - loss: 0.1312 - acc: 0.9612 - val_loss: 0.3004 - val_acc: 0.8813
Epoch 10/20
30/30 [==============================] - 2s 65ms/step - loss: 0.1201 - acc: 0.9656 - val_loss: 0.2891 - val_acc: 0.8866
Epoch 11/20
30/30 [==============================] - 2s 52ms/step - loss: 0.1104 - acc: 0.9698 - val_loss: 0.2941 - val_acc: 0.8852
Epoch 12/20
30/30 [==============================] - 2s 52ms/step - loss: 0.1015 - acc: 0.9725 - val_loss: 0.3013 - val_acc: 0.8835
Epoch 13/20
30/30 [==============================] - 2s 53ms/step - loss: 0.0927 - acc: 0.9762 - val_loss: 0.3103 - val_acc: 0.8812
Epoch 14/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0869 - acc: 0.9783 - val_loss: 0.3166 - val_acc: 0.8811
Epoch 15/20
30/30 [==============================] - 1s 42ms/step - loss: 0.0797 - acc: 0.9806 - val_loss: 0.3337 - val_acc: 0.8770
Epoch 16/20
30/30 [==============================] - 1s 43ms/step - loss: 0.0740 - acc: 0.9834 - val_loss: 0.3353 - val_acc: 0.8812
Epoch 17/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0690 - acc: 0.9853 - val_loss: 0.3456 - val_acc: 0.8768
Epoch 18/20
30/30 [==============================] - 2s 63ms/step - loss: 0.0633 - acc: 0.9870 - val_loss: 0.3516 - val_acc: 0.8791
Epoch 19/20
30/30 [==============================] - 2s 51ms/step - loss: 0.0588 - acc: 0.9891 - val_loss: 0.3605 - val_acc: 0.8779
Epoch 20/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0537 - acc: 0.9904 - val_loss: 0.3866 - val_acc: 0.8701
```

```
history_dict = history.history
history_dict.keys()
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
acc_values = history_dict['acc']
epochs = range(1, len(acc_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()
```

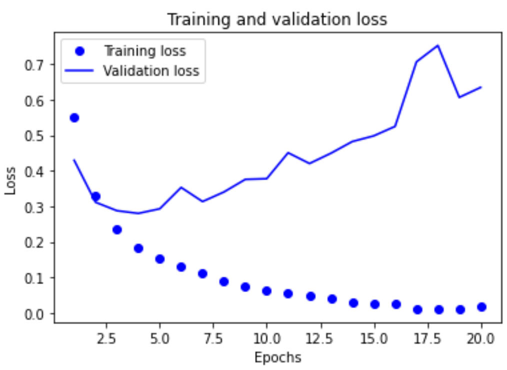


Neural Network with 3 hiden layers

```
model_2 = models.Sequential()
model_2.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model_2.add(layers.Dense(16, activation='relu'))
model_2.add(layers.Dense(16, activation='relu'))
model_2.add(layers.Dense(1, activation='sigmoid'))

model_2.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['acc'])

history = model_2.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

```
Epoch 1/20
30/30 [==============================] - 3s 73ms/step - loss: 0.5520 - acc: 0.7491 - val_loss: 0.4293 - val_acc: 0.8340
Epoch 2/20
30/30 [==============================] - 1s 41ms/step - loss: 0.3284 - acc: 0.8889 - val_loss: 0.3112 - val_acc: 0.8808
Epoch 3/20
30/30 [==============================] - 1s 45ms/step - loss: 0.2363 - acc: 0.9167 - val_loss: 0.2874 - val_acc: 0.8865
Epoch 4/20
30/30 [==============================] - 2s 63ms/step - loss: 0.1834 - acc: 0.9377 - val_loss: 0.2799 - val_acc: 0.8879
Epoch 5/20
30/30 [==============================] - 1s 42ms/step - loss: 0.1545 - acc: 0.9486 - val_loss: 0.2929 - val_acc: 0.8856
Epoch 6/20
30/30 [==============================] - 1s 39ms/step - loss: 0.1325 - acc: 0.9567 - val_loss: 0.3529 - val_acc: 0.8664
Epoch 7/20
30/30 [==============================] - 1s 41ms/step - loss: 0.1103 - acc: 0.9637 - val_loss: 0.3132 - val_acc: 0.8842
Epoch 8/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0901 - acc: 0.9725 - val_loss: 0.3401 - val_acc: 0.8814
Epoch 9/20
30/30 [==============================] - 1s 42ms/step - loss: 0.0763 - acc: 0.9785 - val_loss: 0.3756 - val_acc: 0.8722
Epoch 10/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0645 - acc: 0.9823 - val_loss: 0.3775 - val_acc: 0.8789
Epoch 11/20
30/30 [==============================] - 2s 52ms/step - loss: 0.0548 - acc: 0.9851 - val_loss: 0.4504 - val_acc: 0.8642
Epoch 12/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0484 - acc: 0.9869 - val_loss: 0.4202 - val_acc: 0.8759
Epoch 13/20
30/30 [==============================] - 2s 63ms/step - loss: 0.0400 - acc: 0.9891 - val_loss: 0.4491 - val_acc: 0.8706
Epoch 14/20
30/30 [==============================] - 1s 49ms/step - loss: 0.0303 - acc: 0.9932 - val_loss: 0.4820 - val_acc: 0.8698
Epoch 15/20
30/30 [==============================] - 1s 42ms/step - loss: 0.0260 - acc: 0.9939 - val_loss: 0.4979 - val_acc: 0.8737
Epoch 16/20
30/30 [==============================] - 1s 42ms/step - loss: 0.0252 - acc: 0.9935 - val_loss: 0.5246 - val_acc: 0.8704
Epoch 17/20
30/30 [==============================] - 1s 42ms/step - loss: 0.0125 - acc: 0.9983 - val_loss: 0.7062 - val_acc: 0.8542
Epoch 18/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0112 - acc: 0.9983 - val_loss: 0.7522 - val_acc: 0.8529
Epoch 19/20
30/30 [==============================] - 1s 44ms/step - loss: 0.0116 - acc: 0.9980 - val_loss: 0.6062 - val_acc: 0.8698
Epoch 20/20
30/30 [==============================] - 1s 39ms/step - loss: 0.0177 - acc: 0.9944 - val_loss: 0.6343 - val_acc: 0.8703
```

```
history_dict = history.history
history_dict.keys()
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
acc_values = history_dict['acc']
epochs = range(1, len(acc_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()
```

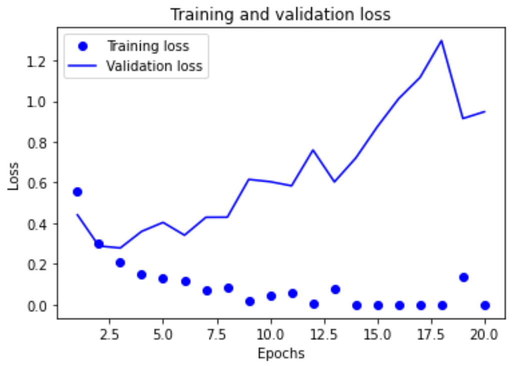


With 5 hidden units and varying neurons in each layer

```
model_3 = models.Sequential()
model_3.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model_3.add(layers.Dense(32, activation='relu'))
model_3.add(layers.Dense(64, activation='relu'))
model_3.add(layers.Dense(128, activation='relu'))
model_3.add(layers.Dense(256, activation='relu'))

model_3.add(layers.Dense(1, activation='sigmoid'))

model_3.compile(optimizer='rmsprop',
```

```
              loss='binary_crossentropy',
              metrics=['acc'])

history = model_3.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
acc_values = history_dict['acc']
epochs = range(1, len(acc_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()
```

```
Epoch 1/20
30/30 [==============================] - 4s 76ms/step - loss: 0.5598 - acc: 0.
Epoch 2/20
30/30 [==============================] - 1s 43ms/step - loss: 0.3021 - acc: 0.
Epoch 3/20
30/30 [==============================] - 1s 44ms/step - loss: 0.2120 - acc: 0.
Epoch 4/20
30/30 [==============================] - 1s 43ms/step - loss: 0.1495 - acc: 0.
Epoch 5/20
30/30 [==============================] - 1s 43ms/step - loss: 0.1306 - acc: 0.
Epoch 6/20
30/30 [==============================] - 1s 45ms/step - loss: 0.1167 - acc: 0.
Epoch 7/20
30/30 [==============================] - 2s 64ms/step - loss: 0.0693 - acc: 0.
Epoch 8/20
30/30 [==============================] - 2s 57ms/step - loss: 0.0880 - acc: 0.
Epoch 9/20
30/30 [==============================] - 1s 44ms/step - loss: 0.0185 - acc: 0.
Epoch 10/20
30/30 [==============================] - 1s 44ms/step - loss: 0.0472 - acc: 0.
Epoch 11/20
30/30 [==============================] - 1s 44ms/step - loss: 0.0571 - acc: 0.
Epoch 12/20
30/30 [==============================] - 1s 43ms/step - loss: 0.0038 - acc: 0.
Epoch 13/20
30/30 [==============================] - 1s 46ms/step - loss: 0.0770 - acc: 0.
Epoch 14/20
30/30 [==============================] - 1s 44ms/step - loss: 0.0026 - acc: 0.
Epoch 15/20
30/30 [==============================] - 1s 44ms/step - loss: 7.7756e-04 - acc
Epoch 16/20
30/30 [==============================] - 2s 67ms/step - loss: 2.9099e-04 - acc
Epoch 17/20
30/30 [==============================] - 2s 60ms/step - loss: 1.3306e-04 - acc
Epoch 18/20
30/30 [==============================] - 1s 45ms/step - loss: 9.4351e-05 - acc
Epoch 19/20
30/30 [==============================] - 1s 45ms/step - loss: 0.1370 - acc: 0.
Epoch 20/20
30/30 [==============================] - 1s 45ms/step - loss: 1.8670e-04 - acc
```

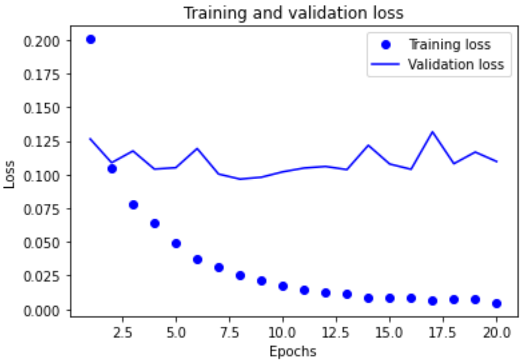


Using MSE loss function

```
model_4 = models.Sequential()
model_4.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model_4.add(layers.Dense(16, activation='relu'))
model_4.add(layers.Dense(1, activation='relu'))
```

```
model_4.compile(optimizer='rmsprop',
              loss='mse',
              metrics=['acc'])

history = model_4.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
acc_values = history_dict['acc']
epochs = range(1, len(acc_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()
```

```
Epoch 1/20
30/30 [==============================] - 4s 101ms/step - loss: 0.2007 - acc: 0
Epoch 2/20
30/30 [==============================] - 1s 39ms/step - loss: 0.1045 - acc: 0.
Epoch 3/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0778 - acc: 0.
Epoch 4/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0639 - acc: 0.
Epoch 5/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0488 - acc: 0.
Epoch 6/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0372 - acc: 0.
Epoch 7/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0315 - acc: 0.
Epoch 8/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0253 - acc: 0.
Epoch 9/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0216 - acc: 0.
Epoch 10/20
30/30 [==============================] - 2s 54ms/step - loss: 0.0172 - acc: 0.
Epoch 11/20
30/30 [==============================] - 2s 53ms/step - loss: 0.0143 - acc: 0.
Epoch 12/20
30/30 [==============================] - 1s 43ms/step - loss: 0.0124 - acc: 0.
Epoch 13/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0116 - acc: 0.
Epoch 14/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0089 - acc: 0.
Epoch 15/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0086 - acc: 0.
Epoch 16/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0083 - acc: 0.
Epoch 17/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0068 - acc: 0.
Epoch 18/20
30/30 [==============================] - 1s 39ms/step - loss: 0.0074 - acc: 0.
Epoch 19/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0076 - acc: 0.
Epoch 20/20
30/30 [==============================] - 1s 50ms/step - loss: 0.0048 - acc: 0.
```

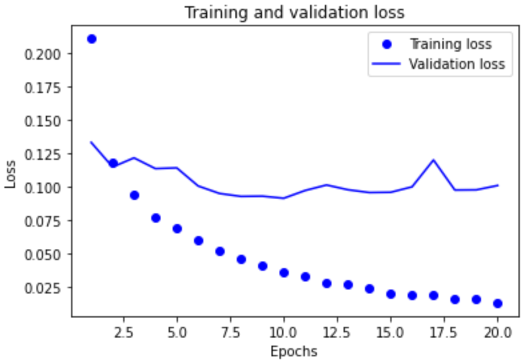


Using tanh activation function

```
model_4 = models.Sequential()
model_4.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
```

```
model_4.add(layers.Dense(16, activation='tanh'))
model_4.add(layers.Dense(1, activation='tanh'))

model_4.compile(optimizer='rmsprop',
              loss='mse',
              metrics=['acc'])

history = model_4.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
acc_values = history_dict['acc']
epochs = range(1, len(acc_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()
```

```
Epoch 1/20
30/30 [==============================] - 4s 113ms/step - loss: 0.2106 - acc: 0
Epoch 2/20
30/30 [==============================] - 3s 88ms/step - loss: 0.1185 - acc: 0.
Epoch 3/20
30/30 [==============================] - 2s 52ms/step - loss: 0.0946 - acc: 0.
Epoch 4/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0770 - acc: 0.
Epoch 5/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0689 - acc: 0.
Epoch 6/20
30/30 [==============================] - 1s 39ms/step - loss: 0.0606 - acc: 0.
Epoch 7/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0526 - acc: 0.
Epoch 8/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0460 - acc: 0.
Epoch 9/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0416 - acc: 0.
Epoch 10/20
30/30 [==============================] - 1s 42ms/step - loss: 0.0368 - acc: 0.
Epoch 11/20
30/30 [==============================] - 2s 64ms/step - loss: 0.0334 - acc: 0.
Epoch 12/20
30/30 [==============================] - 1s 48ms/step - loss: 0.0284 - acc: 0.
Epoch 13/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0274 - acc: 0.
Epoch 14/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0247 - acc: 0.
Epoch 15/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0208 - acc: 0.
Epoch 16/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0192 - acc: 0.
Epoch 17/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0191 - acc: 0.
Epoch 18/20
30/30 [==============================] - 1s 39ms/step - loss: 0.0164 - acc: 0.
Epoch 19/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0160 - acc: 0.
Epoch 20/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0135 - acc: 0.
```



Observation from Task1: Compare the different compiler setups

When I initially used only one hidden layer, I observed that the model had low bias but high variance, indicating that it performed well on the training data but poorly on the test data. To address this, I increased the number of hidden layers to three with equal neurons per layer, but this did not overcome the bias-variance tradeoff. Subsequently, I tried a Neural Network with 5 hidden layers, and while the training loss continued to decrease, the validation loss started increasing, indicating that the model was overfitting the training data and showing an unusual nature in the testing dataset. This may have been due to the limited dataset used during the training process since only partial data from the entire dataset was utilized.

Furthermore, when I compared using mse as a loss function with relu activation and mse loss function with tanh activation, I observed slightly better performance in the second case.

Part 2: Examine the impact of regularization and dropout.

Use the python scripts with fashion_mnist data in HW 3 and testify the impact of adding or without adding the regularization and the impact of adding or without adding the dropout.

Task 1: add the regularization

```
from keras.datasets import fashion_mnist
from keras.utils import np_utils
from keras import regularizers

seed = 7
np.random.seed(seed)

# load data
(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()

# flatten 28*28 images to a 784 vector for each image
num_pixels = X_train.shape[1] * X_train.shape[2]
X_train = X_train.reshape(X_train.shape[0], num_pixels).astype('float32')
X_test = X_test.reshape(X_test.shape[0], num_pixels).astype('float32')

# normalize inputs from 0-255 to 0-1
X_train = X_train / 255
X_test = X_test / 255

# one hot encode outputs
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]
```

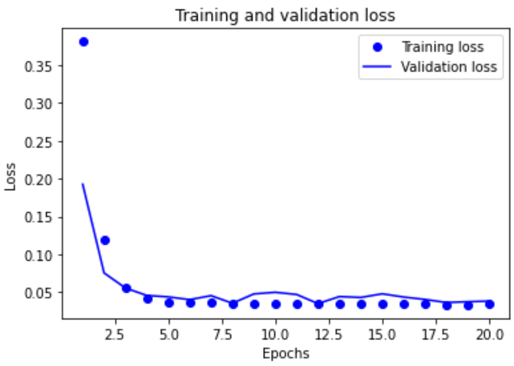
```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 0s 0us/step
```

Task 1: Adding Regularizer

```
network = models.Sequential()
network.add(layers.Dense(512, kernel_regularizer=regularizers.l2(0.001), activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))

network.compile(optimizer='rmsprop',
              loss='mse',
              metrics=['acc'])

history = network.fit(X_train,
                    y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(X_test, y_test))

history_dict = history.history

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
acc_values = history_dict['acc']

epochs = range(1, len(acc_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()
```

```
Epoch 1/20
118/118 [==============================] - 4s 28ms/step - loss: 0.3808 - acc:
Epoch 2/20
118/118 [==============================] - 3s 27ms/step - loss: 0.1199 - acc:
Epoch 3/20
118/118 [==============================] - 4s 32ms/step - loss: 0.0569 - acc:
Epoch 4/20
118/118 [==============================] - 3s 24ms/step - loss: 0.0416 - acc:
Epoch 5/20
118/118 [==============================] - 3s 25ms/step - loss: 0.0371 - acc:
Epoch 6/20
118/118 [==============================] - 3s 26ms/step - loss: 0.0366 - acc:
Epoch 7/20
118/118 [==============================] - 4s 33ms/step - loss: 0.0359 - acc:
Epoch 8/20
118/118 [==============================] - 3s 24ms/step - loss: 0.0357 - acc:
Epoch 9/20
118/118 [==============================] - 3s 24ms/step - loss: 0.0357 - acc:
Epoch 10/20
118/118 [==============================] - 3s 26ms/step - loss: 0.0350 - acc:
Epoch 11/20
118/118 [==============================] - 4s 33ms/step - loss: 0.0349 - acc:
Epoch 12/20
118/118 [==============================] - 3s 25ms/step - loss: 0.0347 - acc:
Epoch 13/20
118/118 [==============================] - 3s 24ms/step - loss: 0.0346 - acc:
Epoch 14/20
118/118 [==============================] - 3s 25ms/step - loss: 0.0346 - acc:
Epoch 15/20
118/118 [==============================] - 4s 34ms/step - loss: 0.0344 - acc:
Epoch 16/20
118/118 [==============================] - 3s 25ms/step - loss: 0.0343 - acc:
Epoch 17/20
118/118 [==============================] - 3s 26ms/step - loss: 0.0342 - acc:
Epoch 18/20
118/118 [==============================] - 3s 25ms/step - loss: 0.0337 - acc:
Epoch 19/20
118/118 [==============================] - 4s 34ms/step - loss: 0.0333 - acc:
Epoch 20/20
118/118 [==============================] - 3s 26ms/step - loss: 0.0350 - acc:
```

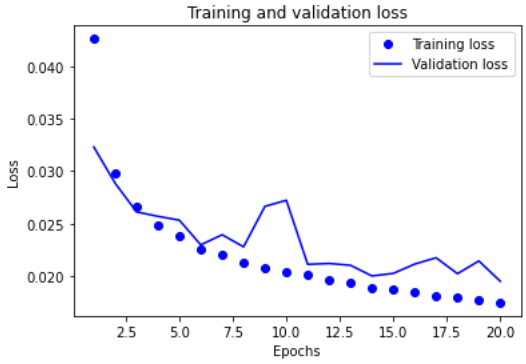


Task 2: Adding Dropout

```
network_dropout = models.Sequential()
network_dropout.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network_dropout.add(layers.Dropout(0.5))
network_dropout.add(layers.Dense(10, activation='softmax'))


network_dropout.compile(optimizer='rmsprop',
              loss='mse',
              metrics=['acc'])

history = network_dropout.fit(X_train,
                    y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(X_test, y_test))

history_dict = history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
acc_values = history_dict['acc']
```

```
epochs = range(1, len(acc_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()
```

```
Epoch 1/20
118/118 [==============================] - 5s 31ms/step - loss: 0.0426 - acc:
Epoch 2/20
118/118 [==============================] - 5s 39ms/step - loss: 0.0298 - acc:
Epoch 3/20
118/118 [==============================] - 3s 29ms/step - loss: 0.0266 - acc:
Epoch 4/20
118/118 [==============================] - 3s 29ms/step - loss: 0.0248 - acc:
Epoch 5/20
118/118 [==============================] - 4s 36ms/step - loss: 0.0238 - acc:
Epoch 6/20
118/118 [==============================] - 4s 30ms/step - loss: 0.0226 - acc:
Epoch 7/20
118/118 [==============================] - 3s 28ms/step - loss: 0.0221 - acc:
Epoch 8/20
118/118 [==============================] - 3s 29ms/step - loss: 0.0213 - acc:
Epoch 9/20
118/118 [==============================] - 5s 40ms/step - loss: 0.0207 - acc:
Epoch 10/20
118/118 [==============================] - 3s 30ms/step - loss: 0.0204 - acc:
Epoch 11/20
118/118 [==============================] - 4s 30ms/step - loss: 0.0201 - acc:
Epoch 12/20
118/118 [==============================] - 4s 36ms/step - loss: 0.0196 - acc:
Epoch 13/20
118/118 [==============================] - 3s 29ms/step - loss: 0.0194 - acc:
Epoch 14/20
118/118 [==============================] - 3s 29ms/step - loss: 0.0188 - acc:
Epoch 15/20
118/118 [==============================] - 3s 29ms/step - loss: 0.0187 - acc:
Epoch 16/20
118/118 [==============================] - 8s 67ms/step - loss: 0.0185 - acc:
Epoch 17/20
118/118 [==============================] - 4s 30ms/step - loss: 0.0181 - acc:
Epoch 18/20
118/118 [==============================] - 4s 37ms/step - loss: 0.0180 - acc:
Epoch 19/20
118/118 [==============================] - 4s 31ms/step - loss: 0.0177 - acc:
Epoch 20/20
118/118 [==============================] - 3s 28ms/step - loss: 0.0175 - acc:
```

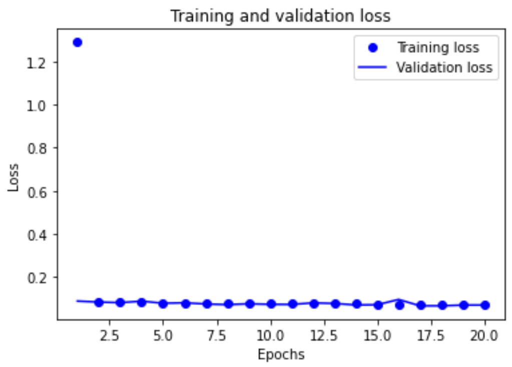


Now Adding both Regularizer and Dropout

```
network_dropout_regularizer = models.Sequential()
network_dropout_regularizer.add(layers.Dense(256, kernel_regularizer=regularizers.l2(0.01), activation='relu', input_shape=(28
network_dropout_regularizer.add(layers.Dropout(0.4))
network_dropout_regularizer.add(layers.Dense(512, kernel_regularizer=regularizers.l2(0.01), activation='relu', input_shape=(28
network_dropout_regularizer.add(layers.Dropout(0.4))

network_dropout_regularizer.add(layers.Dense(10, activation='softmax'))


network_dropout_regularizer.compile(optimizer='rmsprop',
              loss='mse',
              metrics=['acc'])

history = network_dropout_regularizer.fit(X_train,
                    y_train,
                    epochs=20,
```

```
                    batch_size=512,
                    validation_data=(X_test, y_test))

history_dict = history.history

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
acc_values = history_dict['acc']

epochs = range(1, len(acc_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()
```

```
Epoch 1/20
118/118 [==============================] - 6s 34ms/step - loss: 1.2913 - acc:
Epoch 2/20
118/118 [==============================] - 4s 35ms/step - loss: 0.0851 - acc:
Epoch 3/20
118/118 [==============================] - 5s 41ms/step - loss: 0.0825 - acc:
Epoch 4/20
118/118 [==============================] - 4s 31ms/step - loss: 0.0809 - acc:
Epoch 5/20
118/118 [==============================] - 4s 30ms/step - loss: 0.0799 - acc:
Epoch 6/20
118/118 [==============================] - 5s 40ms/step - loss: 0.0785 - acc:
Epoch 7/20
118/118 [==============================] - 4s 30ms/step - loss: 0.0779 - acc:
Epoch 8/20
118/118 [==============================] - 4s 32ms/step - loss: 0.0773 - acc:
Epoch 9/20
118/118 [==============================] - 5s 40ms/step - loss: 0.0771 - acc:
Epoch 10/20
118/118 [==============================] - 4s 31ms/step - loss: 0.0762 - acc:
Epoch 11/20
118/118 [==============================] - 4s 32ms/step - loss: 0.0753 - acc:
Epoch 12/20
118/118 [==============================] - 4s 37ms/step - loss: 0.0752 - acc:
Epoch 13/20
118/118 [==============================] - 4s 38ms/step - loss: 0.0749 - acc:
Epoch 14/20
118/118 [==============================] - 4s 33ms/step - loss: 0.0743 - acc:
Epoch 15/20
118/118 [==============================] - 4s 34ms/step - loss: 0.0736 - acc:
Epoch 16/20
118/118 [==============================] - 5s 39ms/step - loss: 0.0734 - acc:
Epoch 17/20
118/118 [==============================] - 4s 32ms/step - loss: 0.0731 - acc:
Epoch 18/20
118/118 [==============================] - 4s 31ms/step - loss: 0.0726 - acc:
Epoch 19/20
118/118 [==============================] - 5s 42ms/step - loss: 0.0718 - acc:
Epoch 20/20
118/118 [==============================] - 4s 31ms/step - loss: 0.0720 - acc:
```



Part 2: Examine the impact of regularization and dropout.

Observation:

By incorporating regularizers into the layer, it is possible to improve the balance between bias and variance. I assessed the performance of the Neural Network with three different techniques: L2 regularizer, dropout, and a combination of dropout and L2 regularizer. The model with L2 regularizer exhibited good performance in terms of accuracy on the testing dataset compared to the model without any regularization.

Moreover, the addition of dropout helped to mitigate the bias-variance tradeoff. The combination of dropout and L2 regularization shows the best technique in minimizing the bias-variance tradeoff and any overfitting issues in my customized network with tuning parameters.

✓ 1m 25s completed at 11:12 PM