```
!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.15.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.5.26)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (16.0.6)
Requirement already satisfied: ml-dtypes~=0.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.25.2)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/python
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.9.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.36.0
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.60.1)
Requirement already satisfied: tensorboard<2.16,>=2.15 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.2)
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.0)
Requirement already satisfied: keras<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow) (0.42.
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflo
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->te
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (3.
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (3.
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboar
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib<2,>=0.5->t
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboar
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.1
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.16
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.16
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard<2.16,>=2.
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-auth<
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oa
```

```
!pip install keras
```

```
Requirement already satisfied: keras in /usr/local/lib/python3.10/dist-packages (2.15.0)
```

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

```
train_data[0]
```

```
       5,
       144,
       30,
       5535,
       18,
       51,
       36,
       28,
       224,
       92,
       25,
       104,
       4,
       226,
       65,
       16,
       38,
       1334,
       88,
       12,
       16,
       283,
       5,
       16,
       4472,
       113,
       103,
       32,
       15,
       16,
       5345,
       19,
       178,
       32]
```

```
train_labels[0]
```

```
1
```

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

```
9999
```

```
word_index = imdb.get_word_index()
reverse_word_index = dict(
    [(value, key) for (key, value) in word_index.items()])
decoded_review = " ".join(
    [reverse_word_index.get(i - 3, "?") for i in train_data[0]])
```

```
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
x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)
```

```
x_train[0]
```

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

```
y_train = np.asarray(train_labels).astype("float32")
y_test = np.asarray(test_labels).astype("float32")
```

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

Step 1 :

1. Sequential Three layered approach
2. Replaced relu with tanh

3. optimizers changed to adam and loss to mse and metrics == accuracy

```
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense

model = keras.Sequential()
model.add(Dense(16, activation="tanh"))
model.add(Dense(16, activation="tanh"))
model.add(Dense(1, activation="sigmoid"))


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


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 [==============================] - 7s 113ms/step - loss: 0.1818 - accuracy: 0.7774 - val_loss: 0.1267 - val_accuracy: 0.8576
Epoch 2/20
30/30 [==============================] - 1s 36ms/step - loss: 0.0905 - accuracy: 0.9013 - val_loss: 0.0922 - val_accuracy: 0.8826
Epoch 3/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0590 - accuracy: 0.9363 - val_loss: 0.0846 - val_accuracy: 0.8889
Epoch 4/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0418 - accuracy: 0.9581 - val_loss: 0.0825 - val_accuracy: 0.8881
Epoch 5/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0308 - accuracy: 0.9725 - val_loss: 0.0846 - val_accuracy: 0.8848
Epoch 6/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0231 - accuracy: 0.9817 - val_loss: 0.0869 - val_accuracy: 0.8824
Epoch 7/20
30/30 [==============================] - 1s 47ms/step - loss: 0.0177 - accuracy: 0.9872 - val_loss: 0.0899 - val_accuracy: 0.8779
Epoch 8/20
30/30 [==============================] - 2s 51ms/step - loss: 0.0138 - accuracy: 0.9906 - val_loss: 0.0919 - val_accuracy: 0.8795
Epoch 9/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0112 - accuracy: 0.9921 - val_loss: 0.0945 - val_accuracy: 0.8755
Epoch 10/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0093 - accuracy: 0.9930 - val_loss: 0.0971 - val_accuracy: 0.8742
Epoch 11/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0079 - accuracy: 0.9941 - val_loss: 0.0986 - val_accuracy: 0.8735
Epoch 12/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0069 - accuracy: 0.9951 - val_loss: 0.1005 - val_accuracy: 0.8724
Epoch 13/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0061 - accuracy: 0.9954 - val_loss: 0.1013 - val_accuracy: 0.8730
Epoch 14/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0056 - accuracy: 0.9955 - val_loss: 0.1022 - val_accuracy: 0.8721
Epoch 15/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0052 - accuracy: 0.9957 - val_loss: 0.1031 - val_accuracy: 0.8720
Epoch 16/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0049 - accuracy: 0.9959 - val_loss: 0.1040 - val_accuracy: 0.8723
Epoch 17/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0046 - accuracy: 0.9960 - val_loss: 0.1046 - val_accuracy: 0.8713
Epoch 18/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0044 - accuracy: 0.9962 - val_loss: 0.1056 - val_accuracy: 0.8705
Epoch 19/20
30/30 [==============================] - 2s 51ms/step - loss: 0.0043 - accuracy: 0.9962 - val_loss: 0.1061 - val_accuracy: 0.8712
Epoch 20/20
30/30 [==============================] - 1s 49ms/step - loss: 0.0041 - accuracy: 0.9963 - val_loss: 0.1065 - val_accuracy: 0.8713
```

```
# Step 2

### implement dropouts and Regularizers
### check performance by changing the dense layers to 64 hidden units

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


model = keras.Sequential()
model.add(Dense(64, activation="tanh"))
model.add(Dropout(0.5))
model.add(Dense(64, activation="tanh"))
model.add(Dropout(0.5))
model.add(Dense(64, activation="tanh"))
```

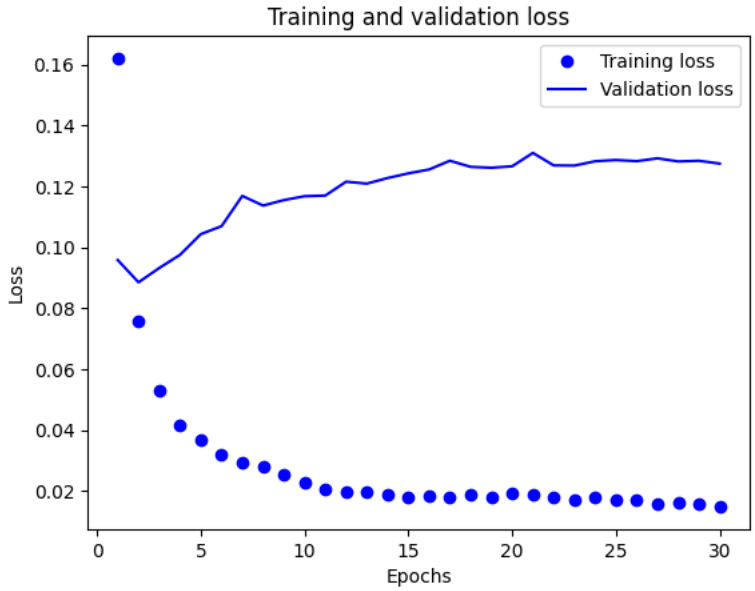
```
model.add(Dense(1, activation="sigmoid",activity_regularizer=regularizers.L2(0.01)))


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

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=30,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

```
Epoch 1/30
30/30 [==============================] - 4s 101ms/step - loss: 0.1621 - accuracy: 0.7662 - val_loss: 0.0958 - val_accuracy: 0.8729
Epoch 2/30
30/30 [==============================] - 2s 67ms/step - loss: 0.0755 - accuracy: 0.9046 - val_loss: 0.0885 - val_accuracy: 0.8869
Epoch 3/30
30/30 [==============================] - 2s 61ms/step - loss: 0.0530 - accuracy: 0.9391 - val_loss: 0.0932 - val_accuracy: 0.8832
Epoch 4/30
30/30 [==============================] - 3s 108ms/step - loss: 0.0415 - accuracy: 0.9545 - val_loss: 0.0975 - val_accuracy: 0.8818
Epoch 5/30
30/30 [==============================] - 2s 70ms/step - loss: 0.0369 - accuracy: 0.9610 - val_loss: 0.1043 - val_accuracy: 0.8772
Epoch 6/30
30/30 [==============================] - 2s 66ms/step - loss: 0.0319 - accuracy: 0.9675 - val_loss: 0.1069 - val_accuracy: 0.8762
Epoch 7/30
30/30 [==============================] - 2s 66ms/step - loss: 0.0291 - accuracy: 0.9718 - val_loss: 0.1169 - val_accuracy: 0.8669
Epoch 8/30
30/30 [==============================] - 2s 68ms/step - loss: 0.0280 - accuracy: 0.9717 - val_loss: 0.1137 - val_accuracy: 0.8727
Epoch 9/30
30/30 [==============================] - 2s 65ms/step - loss: 0.0252 - accuracy: 0.9756 - val_loss: 0.1155 - val_accuracy: 0.8725
Epoch 10/30
30/30 [==============================] - 3s 96ms/step - loss: 0.0227 - accuracy: 0.9795 - val_loss: 0.1168 - val_accuracy: 0.8732
Epoch 11/30
30/30 [==============================] - 2s 62ms/step - loss: 0.0206 - accuracy: 0.9827 - val_loss: 0.1170 - val_accuracy: 0.8735
Epoch 12/30
30/30 [==============================] - 2s 61ms/step - loss: 0.0197 - accuracy: 0.9841 - val_loss: 0.1216 - val_accuracy: 0.8666
Epoch 13/30
30/30 [==============================] - 2s 61ms/step - loss: 0.0197 - accuracy: 0.9830 - val_loss: 0.1209 - val_accuracy: 0.8719
Epoch 14/30
30/30 [==============================] - 2s 66ms/step - loss: 0.0186 - accuracy: 0.9849 - val_loss: 0.1228 - val_accuracy: 0.8693
Epoch 15/30
30/30 [==============================] - 2s 68ms/step - loss: 0.0180 - accuracy: 0.9853 - val_loss: 0.1243 - val_accuracy: 0.8675
Epoch 16/30
30/30 [==============================] - 3s 90ms/step - loss: 0.0183 - accuracy: 0.9855 - val_loss: 0.1256 - val_accuracy: 0.8671
Epoch 17/30
30/30 [==============================] - 2s 65ms/step - loss: 0.0179 - accuracy: 0.9852 - val_loss: 0.1284 - val_accuracy: 0.8630
Epoch 18/30
30/30 [==============================] - 2s 63ms/step - loss: 0.0186 - accuracy: 0.9845 - val_loss: 0.1264 - val_accuracy: 0.8656
Epoch 19/30
30/30 [==============================] - 2s 63ms/step - loss: 0.0179 - accuracy: 0.9852 - val_loss: 0.1261 - val_accuracy: 0.8675
Epoch 20/30
30/30 [==============================] - 2s 67ms/step - loss: 0.0191 - accuracy: 0.9839 - val_loss: 0.1266 - val_accuracy: 0.8663
Epoch 21/30
30/30 [==============================] - 2s 68ms/step - loss: 0.0189 - accuracy: 0.9841 - val_loss: 0.1310 - val_accuracy: 0.8628
Epoch 22/30
30/30 [==============================] - 3s 105ms/step - loss: 0.0177 - accuracy: 0.9854 - val_loss: 0.1269 - val_accuracy: 0.8671
Epoch 23/30
30/30 [==============================] - 2s 65ms/step - loss: 0.0170 - accuracy: 0.9866 - val_loss: 0.1268 - val_accuracy: 0.8675
Epoch 24/30
30/30 [==============================] - 2s 59ms/step - loss: 0.0179 - accuracy: 0.9855 - val_loss: 0.1283 - val_accuracy: 0.8662
Epoch 25/30
30/30 [==============================] - 2s 67ms/step - loss: 0.0169 - accuracy: 0.9867 - val_loss: 0.1287 - val_accuracy: 0.8667
Epoch 26/30
30/30 [==============================] - 2s 62ms/step - loss: 0.0169 - accuracy: 0.9871 - val_loss: 0.1283 - val_accuracy: 0.8670
Epoch 27/30
30/30 [==============================] - 2s 63ms/step - loss: 0.0157 - accuracy: 0.9881 - val_loss: 0.1292 - val_accuracy: 0.8671
Epoch 28/30
30/30 [==============================] - 3s 114ms/step - loss: 0.0161 - accuracy: 0.9878 - val_loss: 0.1282 - val_accuracy: 0.8662
Epoch 29/30
30/30 [==============================] - 2s 68ms/step - loss: 0.0157 - accuracy: 0.9887 - val_loss: 0.1284 - val_accuracy: 0.8665
```

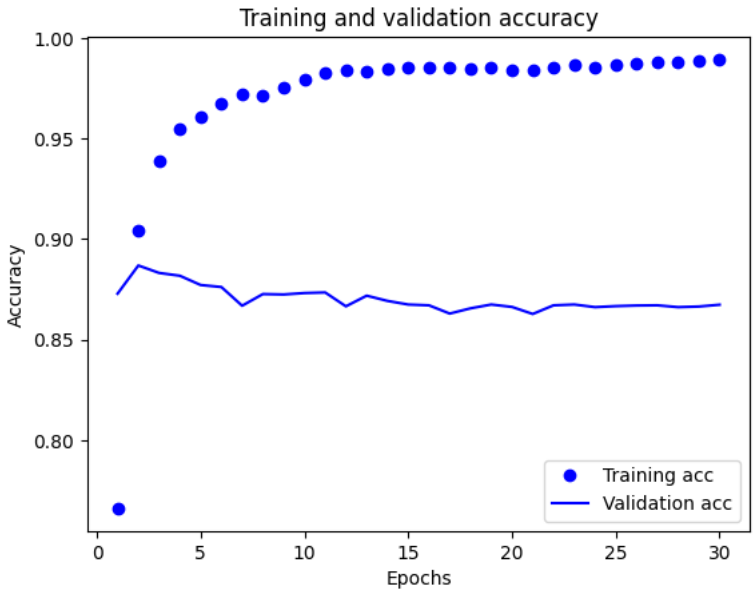
```
history_dict = history.history
history_dict.keys()
```

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

```
import matplotlib.pyplot as plt
history_dict = history.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()
```

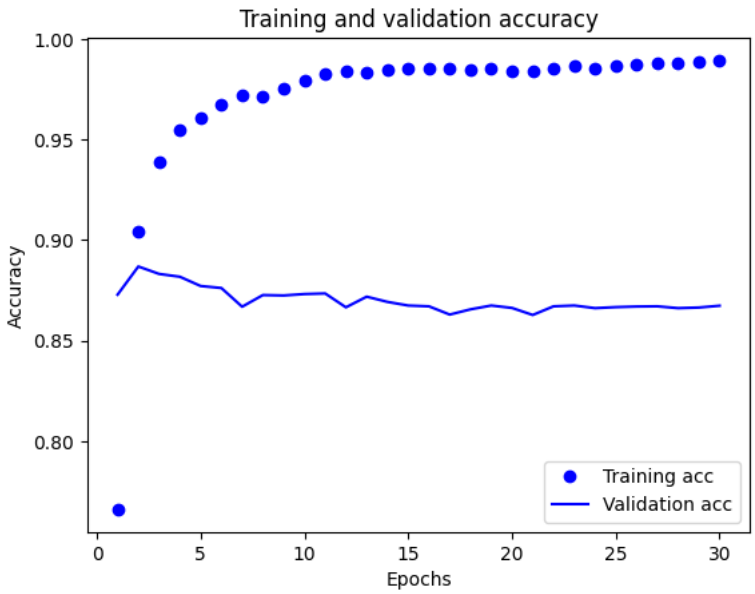



```
plt.clf()
acc = history_dict["accuracy"]
val_acc = history_dict["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()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



Generate Using ... randomly select 5 items from a list Close

```
import matplotlib.pyplot as plt

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()
```

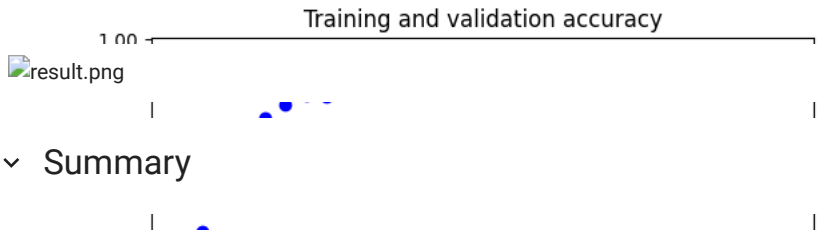



```
from tensorflow.keras.layers import Dropout
from tensorflow.keras import regularizers


model = keras.Sequential()
model.add(Dense(64, activation="tanh"))
model.add(Dropout(0.5))
model.add(Dense(64, activation="tanh"))
model.add(Dense(64, activation="tanh"))
model.add(Dense(64, activation="tanh",activity_regularizer=regularizers.L2(0.01)))
model.add(Dropout(0.5))
model.add(Dense(64, activation="tanh"))
model.add(Dense(64, activation="tanh"))
model.add(Dense(1, activation="sigmoid"))


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

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=30,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

```
Epoch 1/30
30/30 [==============================] - 5s 115ms/step - loss: 0.1748 - accuracy: 0.7837 - val_loss: 0.1062 - val_accuracy: 0.8804
Epoch 2/30
30/30 [==============================] - 2s 75ms/step - loss: 0.0881 - accuracy: 0.9106 - val_loss: 0.0965 - val_accuracy: 0.8804
Epoch 3/30
30/30 [==============================] - 2s 64ms/step - loss: 0.0615 - accuracy: 0.9391 - val_loss: 0.0994 - val_accuracy: 0.8791
Epoch 4/30
30/30 [==============================] - 2s 64ms/step - loss: 0.0504 - accuracy: 0.9485 - val_loss: 0.1024 - val_accuracy: 0.8769
Epoch 5/30
30/30 [==============================] - 2s 69ms/step - loss: 0.0413 - accuracy: 0.9596 - val_loss: 0.1036 - val_accuracy: 0.8744
Epoch 6/30
```

```
30/30 [==============================] - 2s 65ms/step - loss: 0.0351 - accuracy: 0.9651 - val_loss: 0.1057 - val_accuracy: 0.8748
Epoch 7/30
30/30 [==============================] - 3s 117ms/step - loss: 0.0304 - accuracy: 0.9706 - val_loss: 0.1092 - val_accuracy: 0.8721
Epoch 8/30
30/30 [==============================] - 2s 69ms/step - loss: 0.0283 - accuracy: 0.9707 - val_loss: 0.1127 - val_accuracy: 0.8700
Epoch 9/30
30/30 [==============================] - 2s 69ms/step - loss: 0.0261 - accuracy: 0.9747 - val_loss: 0.1171 - val_accuracy: 0.8678
Epoch 10/30
30/30 [==============================] - 2s 64ms/step - loss: 0.0236 - accuracy: 0.9770 - val_loss: 0.1175 - val_accuracy: 0.8690
Epoch 11/30
30/30 [==============================] - 2s 69ms/step - loss: 0.0228 - accuracy: 0.9777 - val_loss: 0.1199 - val_accuracy: 0.8666
Epoch 12/30
30/30 [==============================] - 3s 91ms/step - loss: 0.0209 - accuracy: 0.9797 - val_loss: 0.1195 - val_accuracy: 0.8669
Epoch 13/30
30/30 [==============================] - 2s 73ms/step - loss: 0.0205 - accuracy: 0.9801 - val_loss: 0.1211 - val_accuracy: 0.8676
Epoch 14/30
30/30 [==============================] - 2s 70ms/step - loss: 0.0212 - accuracy: 0.9789 - val_loss: 0.1236 - val_accuracy: 0.8682
Epoch 15/30
30/30 [==============================] - 2s 70ms/step - loss: 0.0199 - accuracy: 0.9806 - val_loss: 0.1222 - val_accuracy: 0.8671
Epoch 16/30
30/30 [==============================] - 2s 66ms/step - loss: 0.0188 - accuracy: 0.9816 - val_loss: 0.1235 - val_accuracy: 0.8649
Epoch 17/30
30/30 [==============================] - 2s 69ms/step - loss: 0.0164 - accuracy: 0.9845 - val_loss: 0.1254 - val_accuracy: 0.8639
Epoch 18/30
30/30 [==============================] - 3s 114ms/step - loss: 0.0187 - accuracy: 0.9811 - val_loss: 0.1268 - val_accuracy: 0.8615
Epoch 19/30
30/30 [==============================] - 2s 70ms/step - loss: 0.0176 - accuracy: 0.9824 - val_loss: 0.1271 - val_accuracy: 0.8637
Epoch 20/30
30/30 [==============================] - 2s 69ms/step - loss: 0.0164 - accuracy: 0.9839 - val_loss: 0.1237 - val_accuracy: 0.8671
Epoch 21/30
30/30 [==============================] - 2s 69ms/step - loss: 0.0161 - accuracy: 0.9845 - val_loss: 0.1272 - val_accuracy: 0.8634
Epoch 22/30
30/30 [==============================] - 2s 70ms/step - loss: 0.0164 - accuracy: 0.9839 - val_loss: 0.1245 - val_accuracy: 0.8664
Epoch 23/30
30/30 [==============================] - 3s 93ms/step - loss: 0.0145 - accuracy: 0.9863 - val_loss: 0.1256 - val_accuracy: 0.8658
Epoch 24/30
30/30 [==============================] - 2s 79ms/step - loss: 0.0150 - accuracy: 0.9853 - val_loss: 0.1312 - val_accuracy: 0.8608
Epoch 25/30
30/30 [==============================] - 2s 67ms/step - loss: 0.0168 - accuracy: 0.9834 - val_loss: 0.1282 - val_accuracy: 0.8640
Epoch 26/30
30/30 [==============================] - 2s 65ms/step - loss: 0.0154 - accuracy: 0.9847 - val_loss: 0.1285 - val_accuracy: 0.8620
Epoch 27/30
30/30 [==============================] - 2s 70ms/step - loss: 0.0152 - accuracy: 0.9845 - val_loss: 0.1286 - val_accuracy: 0.8620
Epoch 28/30
30/30 [==============================] - 2s 63ms/step - loss: 0.0142 - accuracy: 0.9863 - val_loss: 0.1285 - val_accuracy: 0.8655
Epoch 29/30
30/30 [==============================] - 3s 114ms/step - loss: 0.0150 - accuracy: 0.9857 - val_loss: 0.1278 - val_accuracy: 0.8648
```

```
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()
```



result.png

Summary

My approach to tackling the problem involved initially understanding the significance of the Keras Sequential model, which serves as a stack of layers for constructing neural networks. This entailed importing essential modules such as layers, Dense, Dropout, and Regularizers from TensorFlow.keras.

I experimented with building neural networks containing 2, 3, and 6 layers, each with different numbers of hidden neurons (16, 64, and 64 respectively) to assess performance. An important observation was that regardless of the number of layers stacked, the performance plateaued once a certain threshold was reached.

Initializing the Sequential model with model = keras.Sequential() sets up the structure comprising input, hidden, and output layers. Adding a hidden layer with 64 dense units and using the tanh activation function (model.add(Dense(64, activation="tanh")) means creating 64 neurons in the layer to learn vector data.

The Dropout layer (model.add(Dropout(0.5))) plays a crucial role in combating overfitting by randomly dropping out neurons. Specifying 0.5 implies dropping out 50% of the neurons.

Although I experimented with L1 and L2 regularizers, they didn't significantly improve performance and may have even diminished it, suggesting the model may have reached saturation. The best validation accuracy achieved was around 86-87%.

Replacing binary_crossentropy with mean squared error (MSE) for loss evaluation resulted in better performance metrics, with MSE yielding a lower validation loss compared to binary_crossentropy.

ReLU emerged as the preferred activation function over sigmoid and tanh due to its mitigation of the vanishing gradient problem. However, in this context, tanh performed similarly to ReLU.