Assignment 7: Neural Networks using Keras and Tensorflow Please see the associated document for questions

If you have problems with Keras and Tensorflow on your local installation please make sure they are updated. On Google Colab this notebook runs.

```
pip install tensorflow
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: tensorflow in /usr/local/lib/python3.8/dist-packages (2.11.0)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (0.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (4.5.0)
Requirement already satisfied: flatbuffers>=2.0 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (23.1.21)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (1.15.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (15.0.6.1)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: keras<2.12,>=2.11.0 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (2.11.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.8/dist-packages (from tensorflow) (23.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.8/dist-packages (from tensorflow) (57.4.0)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (3.19.6)
Requirement already satisfied: tensorboard<2.12,>=2.11 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (2.11.2)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (1.22.4)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (0.
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (1.51.3)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (1.15.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (3.1.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (2.2.0)
Requirement already satisfied: tensorflow-estimator<2.12,>=2.11.0 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (2.11
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.8/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.8/dist-packages (from astunparse>=1.6.0->tensorflow) (0
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.8/dist-packages (from tensorboard<2.12,>=2.1
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from tensorboard<2.12,>=2.11->tensorflow)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/python3.8/dist-packages (from tensorboard<2.
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.8/dist-packages (from tensorboard<2.12,>=2.11->tensorf
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.8/dist-packages (from tensorboard<2.12,>=2.11->tenso
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.8/dist-packages (from tensorboard<2.12,>=
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.8/dist-packages (from tensorboard<2.12,>=2.11->tensorflow)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.8/dist-packages (from google-auth<3,>=1.6.3->tensor
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.8/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.1
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.8/dist-packages (from google-auth<3,>=1.6.3->tensorb
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.8/dist-packages (from google-auth-oauthlib<0.5,>=
Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/python3.8/dist-packages (from markdown>=2.6.8->tensorboard
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests<3,>=2.21.0->tensorboard<2.12,>
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests<3,>=2.21.0->tensorboard<
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from requests<3,>=2.21.0->tensorboa
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.8/dist-packages (from requests<3,>=2.21.0->tensorboard<2
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.8/dist-packages (from werkzeug>=1.0.1->tensorboard<2.12,
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.8/dist-packages (from importlib-metadata>=4.4->markdown>=2.6.8->
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.8/dist-packages (from pyasn1-modules>=0.2.1->google-a
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.8/dist-packages (from requests-oauthlib>=0.7.0->google-aut
```

```
# imports
from __future__ import print_function
import keras
from keras import utils as np_utils
import tensorflow
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
import tensorflow as tf
from matplotlib import pyplot as plt
```

```
# Hyper-parameters data-loading and formatting

batch_size = 128
num_classes = 10
epochs = 40

img_rows, img_cols = 28, 28

(x_train, lbl_train), (x_test, lbl_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
```

```
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
```

Preprocessing

```
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')

x_train /= 255
x_test /= 255

y_train = keras.utils.np_utils.to_categorical(lbl_train, num_classes)
y_test = keras.utils.np_utils.to_categorical(lbl_test, num_classes)
```

```
## Define model ##
model = Sequential()

model.add(Flatten())
model.add(Dense(500, activation = 'relu'))
model.add(Dense(300, activation = 'relu'))
model.add(Dense(num_classes, activation='softmax'))
```

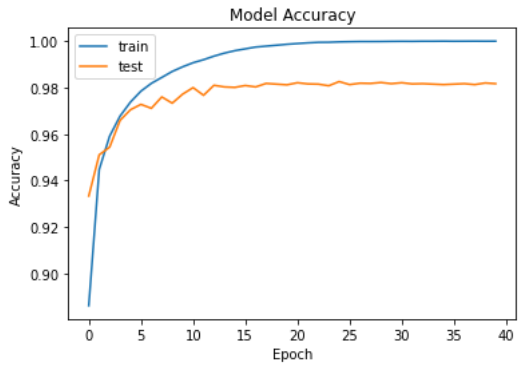
```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=tensorflow.keras.optimizers.SGD(learning_rate = 0.1),
        metrics=['accuracy'],)

fit_info = model.fit(x_train, y_train,
           batch_size=batch_size,
           epochs=epochs,
           verbose=1,
           validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss: {}, Test accuracy {}'.format(score[0], score[1]))
```

```
469/469 [==============================] - 6s 13ms/step - loss: 0.0307 - accuracy: 0.9920 - val_loss: 0.0717 - val_accuracy: 0.97
Epoch 13/40
469/469 [==============================] - 7s 15ms/step - loss: 0.0265 - accuracy: 0.9935 - val_loss: 0.0618 - val_accuracy: 0.98
Epoch 14/40
469/469 [==============================] - 6s 13ms/step - loss: 0.0232 - accuracy: 0.9948 - val_loss: 0.0608 - val_accuracy: 0.98
Epoch 15/40
469/469 [==============================] - 7s 15ms/step - loss: 0.0200 - accuracy: 0.9959 - val_loss: 0.0610 - val_accuracy: 0.98
Epoch 16/40
469/469 [==============================] - 6s 13ms/step - loss: 0.0175 - accuracy: 0.9967 - val_loss: 0.0602 - val_accuracy: 0.98
Epoch 17/40
469/469 [==============================] - 9s 18ms/step - loss: 0.0152 - accuracy: 0.9974 - val_loss: 0.0625 - val_accuracy: 0.98
Epoch 18/40
469/469 [==============================] - 8s 17ms/step - loss: 0.0134 - accuracy: 0.9978 - val_loss: 0.0594 - val_accuracy: 0.98
Epoch 19/40
469/469 [==============================] - 6s 13ms/step - loss: 0.0119 - accuracy: 0.9982 - val_loss: 0.0601 - val_accuracy: 0.98
Epoch 20/40
469/469 [==============================] - 7s 16ms/step - loss: 0.0103 - accuracy: 0.9986 - val_loss: 0.0599 - val_accuracy: 0.98
Epoch 21/40
```

```
Epoch 33/40
469/469 [==============================] - 7s 15ms/step - loss: 0.0029 - accuracy: 0.9999 - val_loss: 0.0629 - val_accuracy: 0.98
Epoch 34/40
469/469 [==============================] - 7s 14ms/step - loss: 0.0027 - accuracy: 0.9999 - val_loss: 0.0638 - val_accuracy: 0.98
Epoch 35/40
469/469 [==============================] - 8s 17ms/step - loss: 0.0025 - accuracy: 1.0000 - val_loss: 0.0640 - val_accuracy: 0.98
Epoch 36/40
469/469 [==============================] - 6s 14ms/step - loss: 0.0023 - accuracy: 0.9999 - val_loss: 0.0645 - val_accuracy: 0.98
Epoch 37/40
469/469 [==============================] - 7s 16ms/step - loss: 0.0022 - accuracy: 1.0000 - val_loss: 0.0639 - val_accuracy: 0.98
Epoch 38/40
469/469 [==============================] - 6s 14ms/step - loss: 0.0021 - accuracy: 1.0000 - val_loss: 0.0651 - val_accuracy: 0.98
Epoch 39/40
469/469 [==============================] - 7s 16ms/step - loss: 0.0020 - accuracy: 1.0000 - val_loss: 0.0641 - val_accuracy: 0.98
Epoch 40/40
469/469 [==============================] - 6s 13ms/step - loss: 0.0019 - accuracy: 1.0000 - val_loss: 0.0660 - val_accuracy: 0.98
```

```
# Plot accuracy
plt.plot(fit_info.history['accuracy'])
plt.plot(fit_info.history['val_accuracy'])
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```




```
def train_using_decay(decay):
    model = Sequential()

    model.add(Flatten())
    model.add(Dense(500, activation = 'relu', kernel_regularizer=tf.keras.regularizers.l2(decay)))
    model.add(Dense(300, activation = 'relu', kernel_regularizer=tf.keras.regularizers.l2(decay)))
    model.add(Dense(num_classes, activation='softmax'))


    model.compile(loss=keras.losses.categorical_crossentropy,
                optimizer=tensorflow.keras.optimizers.SGD(learning_rate = 0.1),
          metrics=['accuracy'],)

    fit_info = model.fit(x_train, y_train,
            batch_size=batch_size,
            epochs=40,
            verbose=1,
            validation_data=(x_test, y_test))
    score = model.evaluate(x_test, y_test, verbose=0)
    print('Test loss: {}, Test accuracy {}'.format(score[0], score[1]))
    return fit_info


# train for 3 regularization
decays = [0.000001, 0.00001, 0.001]
info = []
for i in decays:
    print("[*] Training with {} decay".format(i))
    info.append(train_using_decay(i))
```

```
469/469 [==============================] - 8s 18ms/step - loss: 0.1514 - accuracy: 0.9884 - val_loss: 0.1686 - val_accuracy: 0.97
Epoch 19/40
469/469 [==============================] - 6s 13ms/step - loss: 0.1437 - accuracy: 0.9881 - val_loss: 0.1625 - val_accuracy: 0.97
Epoch 20/40
469/469 [==============================] - 7s 15ms/step - loss: 0.1362 - accuracy: 0.9894 - val_loss: 0.1594 - val_accuracy: 0.97
Epoch 21/40
469/469 [==============================] - 6s 13ms/step - loss: 0.1300 - accuracy: 0.9894 - val_loss: 0.1511 - val_accuracy: 0.97
Epoch 22/40
469/469 [==============================] - 7s 14ms/step - loss: 0.1250 - accuracy: 0.9900 - val_loss: 0.1495 - val_accuracy: 0.97
Epoch 23/40
469/469 [==============================] - 6s 13ms/step - loss: 0.1207 - accuracy: 0.9905 - val_loss: 0.1433 - val_accuracy: 0.98
Epoch 24/40
469/469 [==============================] - 7s 15ms/step - loss: 0.1167 - accuracy: 0.9902 - val_loss: 0.1398 - val_accuracy: 0.98
Epoch 25/40
469/469 [==============================] - 6s 13ms/step - loss: 0.1141 - accuracy: 0.9905 - val_loss: 0.1338 - val_accuracy: 0.98
Epoch 26/40
469/469 [==============================] - 7s 14ms/step - loss: 0.1104 - accuracy: 0.9909 - val_loss: 0.1335 - val_accuracy: 0.98
Epoch 27/40
469/469 [==============================] - 6s 13ms/step - loss: 0.1081 - accuracy: 0.9912 - val_loss: 0.1399 - val_accuracy: 0.97
Epoch 28/40
469/469 [==============================] - 7s 15ms/step - loss: 0.1063 - accuracy: 0.9908 - val_loss: 0.1315 - val_accuracy: 0.97
Epoch 29/40
469/469 [==============================] - 6s 13ms/step - loss: 0.1037 - accuracy: 0.9918 - val_loss: 0.1373 - val_accuracy: 0.97
Epoch 30/40
469/469 [==============================] - 7s 14ms/step - loss: 0.1020 - accuracy: 0.9919 - val_loss: 0.1319 - val_accuracy: 0.97
Epoch 31/40
469/469 [==============================] - 6s 13ms/step - loss: 0.1005 - accuracy: 0.9920 - val_loss: 0.1276 - val_accuracy: 0.98
Epoch 32/40
469/469 [==============================] - 7s 15ms/step - loss: 0.0983 - accuracy: 0.9923 - val_loss: 0.1389 - val_accuracy: 0.97
Epoch 33/40
469/469 [==============================] - 6s 13ms/step - loss: 0.0976 - accuracy: 0.9927 - val_loss: 0.1280 - val_accuracy: 0.97
Epoch 34/40
469/469 [==============================] - 6s 14ms/step - loss: 0.0960 - accuracy: 0.9923 - val_loss: 0.1287 - val_accuracy: 0.97
Epoch 35/40
469/469 [==============================] - 6s 13ms/step - loss: 0.0951 - accuracy: 0.9926 - val_loss: 0.1198 - val_accuracy: 0.98
Epoch 36/40
469/469 [==============================] - 7s 15ms/step - loss: 0.0936 - accuracy: 0.9927 - val_loss: 0.1181 - val_accuracy: 0.98
Epoch 37/40
469/469 [==============================] - 6s 13ms/step - loss: 0.0927 - accuracy: 0.9931 - val_loss: 0.1212 - val_accuracy: 0.98
Epoch 38/40
469/469 [==============================] - 7s 15ms/step - loss: 0.0921 - accuracy: 0.9926 - val_loss: 0.1210 - val_accuracy: 0.98
Epoch 39/40
469/469 [==============================] - 6s 13ms/step - loss: 0.0911 - accuracy: 0.9931 - val_loss: 0.1201 - val_accuracy: 0.97
Epoch 40/40
```

```
hist.history['val_accuracy']
```

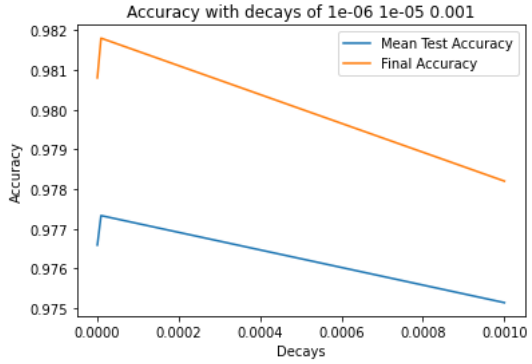
```
[0.9340000152587891,
 0.9495999813079834,
 0.9613000154495239,
 0.964900016784668,
 0.9678000211715698,
 0.9682999849319458,
 0.9728999733924866,
 0.9740999937057495,
 0.9733999967575073,
 0.9758999943733215,
 0.973800003528595,
 0.9739000201225281,
 0.9761999845504761,
 0.9764000177383423,
 0.9772999882698059,
 0.9764000177383423,
 0.9778000116348267,
 0.9793000221252441,
 0.979200005531311,
 0.9789999723434448,
 0.979200005531311,
 0.9787999987602234,
 0.9801999926567078,
 0.9815000295639038,
 0.9819999933242798,
 0.980400025844574,
 0.9785000085830688,
 0.9797000288963318,
 0.9768999814987183,
 0.9790999889373779,
 0.9801999926567078,
 0.9763000011444092,
 0.9799000024795532,
 0.9779999852180481,
 0.9824000000953674,
 0.9824000000953674,
 0.9800000190734863,
 0.9804999828338623,
 0.9799000024795532,
 0.9782000184059143]
```

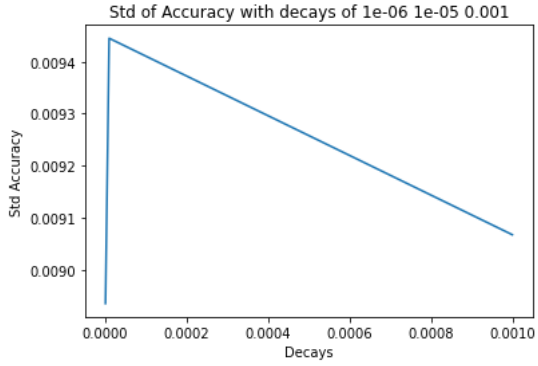
```
#  Plot the final validation accuracy with standard deviation (computed from the replicates) as a function of the regularization factor
i = 0

x = []
y_mean_acc = []
y_std_acc = []
y_acc = []
for hist in info:
    x.append(decays[i])
    y_mean_acc.append(np.mean(hist.history['val_accuracy'])) # get the mean accuracy of the model
    y_std_acc.append(np.std(hist.history['val_accuracy']))   # get the std of the acc
    y_acc.append(hist.history['val_accuracy'][-1])
    i+=1
print(y_acc)
print(y_mean_acc)
print(y_std_acc)
plt.plot(x, y_mean_acc)
plt.plot(x, y_acc)
#plt.plot(x, y_std_acc)
plt.legend(['Mean Test Accuracy', 'Final Accuracy', 'Std Test Accuracy {}'])
plt.title("Accuracy with decays of {} {} {}".format(decays[0], decays[1], decays[2]))
plt.xlabel("Decays")
plt.ylabel("Accuracy")
plt.show()

plt.plot(x, y_std_acc)
plt.title("Std of Accuracy with decays of {} {} {}".format(decays[0], decays[1], decays[2]))
plt.xlabel("Decays")
plt.ylabel("Std Accuracy")
plt.show()
```

```
[0.9807999730110168, 0.9818000197410583, 0.9782000184059143]
[0.9765900000929832, 0.9773324981331826, 0.9751400023698806]
[0.008935263914218573, 0.00944495393084238, 0.009067408224511155]
```






```
def train_conv():
    model = Sequential()

  #model.add(Flatten())
  # Multiple Conv2D layer
    model.add(Conv2D(32, kernel_size=3, padding="valid", input_shape=(28, 28, 1), activation = 'relu')),
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(1, 1))),
    model.add(Conv2D(64, kernel_size=3, padding="valid", activation = 'relu')),
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(1, 1))),
    model.add(Conv2D(100, kernel_size=3, padding="valid", activation = 'relu')),
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(1, 1))),
    model.add(Flatten()),
    model.add(Dense(200, activation='relu')),
    model.add(Dense(num_classes, activation='softmax'))

    model.compile(loss=keras.losses.categorical_crossentropy,
                  optimizer="adam",
          metrics=['accuracy'],)
```

```
    fit_info = model.fit(x_train, y_train,
            batch_size=batch_size,
            epochs=10,
            verbose=1,
            validation_data=(x_test, y_test))
    score = model.evaluate(x_test, y_test, verbose=0)
    print('Test loss: {}, Test accuracy {}'.format(score[0], score[1]))
    return fit_info


train_conv()
```