```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import numpy as np
import pandas as pd
from sklearn.utils import shuffle
import matplotlib.pyplot as plt

#read data from hw3 solution
def read_data(filename='spambase.csv', d=57):
    # read data
    data_df = pd.read_csv(filename, header=None).dropna(axis=0)
    # preprocess
    X = np.array(data_df.iloc[:,:d])
    X = (X-np.mean(X, axis=0))/np.std(X, axis=0) # normalize
    y = np.array(data_df.iloc[:,d:])
    y = y.flatten()
    # seperate classes
    X1 = X[y==1]
    X0 = X[y==0]
    y1 = y[y==1]
    y0 = y[y==0]
    return X0, y0, X1, y1

#train test split from hw3 solution
def train_test_split(X0, y0, X1, y1, split_percent):
    # splitting id
    split_id0 = int(X0.shape[0]*split_percent*.01)
    split_id1 = int(X1.shape[0]*split_percent*.01)
    # random shuffle each class
    X00, y00 = shuffle(X0, y0)
    X11, y11 = shuffle(X1, y1)
    # train set

    Xtrain = np.concatenate((X00[:split_id0], X11[:split_id1]), axis=0)
    ytrain = np.concatenate((y00[:split_id0], y11[:split_id1]), axis=0)
    # test set
    Xtest = np.concatenate((X00[split_id0:], X11[split_id1:]), axis=0)
    ytest = np.concatenate((y00[split_id0:], y11[split_id1:]), axis=0)
    # random shuffle train set
    Xtrain, ytrain = shuffle(Xtrain, ytrain)
    return Xtrain, ytrain, Xtest, ytest

#train data from hw3 solution
def train_data(X, y, train_percent):
    N = X.shape[0]
    split_id = int(N*train_percent/100)
    return X[:split_id], y[:split_id]

def one_10_tanh(filename, num_splits, train_percent):
    #parse
    X0, y0, X1, y1 = read_data(filename)
    # init error
    test_errors = np.zeros((num_splits, len(train_percent)))
    #creat 1 hidden 10 neuron SGD model
```

```
    model = tf.keras.models.Sequential()
    model.add(tf.keras.layers.Dense(10,input_dim = len(X0[0,:]),activation='tanh'))
    model.add(tf.keras.layers.Dense(1,activation = "sigmoid"))
    model.compile(loss = 'binary_crossentropy', optimizer = 'SGD',metrics = ['accuracy'])

    split_percent=80
    for i in range(num_splits):
        Xtrain_i, ytrain_i, Xtest_i, ytest_i = train_test_split(X0, y0, X1, y1,split_percent)
        # try for differnt train set sizes
        for j, percent in enumerate(train_percent):
            Xtrain_ij, ytrain_ij = train_data(Xtrain_i, ytrain_i, percent)
            model.fit(Xtrain_ij,ytrain_ij,epochs=2,batch_size=20,verbose=0)
            loss, acc = model.evaluate(Xtest_i,ytest_i)

            test_errors[i][j] = 1-acc

    test2 = np.zeros((num_splits,len(train_percent)))
    #create 1 hidden, 10 neuron Adam Model
    model2= tf.keras.models.Sequential()
    model2.add(tf.keras.layers.Dense(10,input_dim = len(X0[0,:]),activation='tanh'))
    model2.add(tf.keras.layers.Dense(1,activation = "sigmoid"))
    model2.compile(loss = 'binary_crossentropy', optimizer = 'Adam',metrics = ['accuracy'])
    for i in range(num_splits):
        Xtrain_i, ytrain_i, Xtest_i, ytest_i = train_test_split(X0, y0, X1, y1,split_percent)
        # try for differnt train set sizes
        for j, percent in enumerate(train_percent):
            #create training percent data
            Xtrain_ij, ytrain_ij = train_data(Xtrain_i, ytrain_i, percent)
            #train
            model2.fit(Xtrain_ij,ytrain_ij,epochs=2,batch_size=20,verbose=0)
            #test
            loss, acc = model.evaluate(Xtest_i,ytest_i)
            #store results
            test2[i][j] = 1-acc
    return test_errors,test2

def one_30_relu(filename, num_splits, train_percent):
    #parse
    X0, y0, X1, y1 = read_data(filename)
    # init error
    test_errors = np.zeros((num_splits, len(train_percent)))
    #create 1 hidden 30 neuron SGD model
    model = tf.keras.models.Sequential()
    model.add(tf.keras.layers.Dense(30,input_dim = len(X0[0,:]),activation='relu'))
    model.add(tf.keras.layers.Dense(1,activation = "sigmoid"))
    model.compile(loss = 'binary_crossentropy', optimizer = 'SGD',metrics = ['accuracy'])

    split_percent=80
    for i in range(num_splits):
        Xtrain_i, ytrain_i, Xtest_i, ytest_i = train_test_split(X0, y0, X1, y1,split_percent)
        # try for differnt train set sizes
        for j, percent in enumerate(train_percent):
            Xtrain_ij, ytrain_ij = train_data(Xtrain_i, ytrain_i, percent)
            model.fit(Xtrain_ij,ytrain_ij,epochs=2,batch_size=20,verbose=0)
            loss, acc = model.evaluate(Xtest_i,ytest_i)
```

```
            #print(loss)
            test_errors[i][j] = 1-acc
    print("halfway")
    test2 = np.zeros((num_splits,len(train_percent)))
    model2= tf.keras.models.Sequential()
    model2.add(tf.keras.layers.Dense(30,input_dim = len(X0[0,:]),activation='relu'))
    model2.add(tf.keras.layers.Dense(1,activation = "sigmoid"))
    model2.compile(loss = 'binary_crossentropy', optimizer = 'Adam',metrics = ['accuracy'])
    for i in range(num_splits):
        Xtrain_i, ytrain_i, Xtest_i, ytest_i = train_test_split(X0, y0, X1, y1,split_percent)
        # try for differnt train set sizes
        for j, percent in enumerate(train_percent):
            Xtrain_ij, ytrain_ij = train_data(Xtrain_i, ytrain_i, percent)
            model2.fit(Xtrain_ij,ytrain_ij,epochs=2,batch_size=20,verbose=0)
            loss, acc = model.evaluate(Xtest_i,ytest_i)
            #print(loss)
            test2[i][j] = 1-acc
    return test_errors,test2

def one_10_relu(filename, num_splits, train_percent):
    #parse
    X0, y0, X1, y1 = read_data(filename)
    # init error
    test_errors = np.zeros((num_splits, len(train_percent)))
    #create 1 hidden 10 neuron relu model
    model = tf.keras.models.Sequential()
    model.add(tf.keras.layers.Dense(10,input_dim = len(X0[0,:]),activation='relu'))
    model.add(tf.keras.layers.Dense(1,activation = "sigmoid"))
    model.compile(loss = 'binary_crossentropy', optimizer = 'SGD',metrics = ['accuracy'])

    split_percent=80
    for i in range(num_splits):
        Xtrain_i, ytrain_i, Xtest_i, ytest_i = train_test_split(X0, y0, X1, y1,split_percent)
        # try for differnt train set sizes
        for j, percent in enumerate(train_percent):
            Xtrain_ij, ytrain_ij = train_data(Xtrain_i, ytrain_i, percent)
            model.fit(Xtrain_ij,ytrain_ij,epochs=2,batch_size=20,verbose=0)
            loss, acc = model.evaluate(Xtest_i,ytest_i)
            #print(loss)
            test_errors[i][j] = 1-acc

    test2 = np.zeros((num_splits,len(train_percent)))
    #create 1 hidden 10 neuron Adam model
    model2= tf.keras.models.Sequential()
    model2.add(tf.keras.layers.Dense(10,input_dim = len(X0[0,:]),activation='relu'))
    model2.add(tf.keras.layers.Dense(1,activation = "sigmoid"))
    model2.compile(loss = 'binary_crossentropy', optimizer = 'Adam',metrics = ['accuracy'])
    for i in range(num_splits):
        Xtrain_i, ytrain_i, Xtest_i, ytest_i = train_test_split(X0, y0, X1, y1,split_percent)
        # try for differnt train set sizes
        for j, percent in enumerate(train_percent):
            Xtrain_ij, ytrain_ij = train_data(Xtrain_i, ytrain_i, percent)
            model2.fit(Xtrain_ij,ytrain_ij,epochs=2,batch_size=20,verbose=0)
            loss, acc = model.evaluate(Xtest_i,ytest_i)
            #print(loss)
```

```
            test2[i][j] = 1-acc
    return test_errors,test2

def two_10_relu(filename, num_splits, train_percent):
    #parse
    X0, y0, X1, y1 = read_data(filename)

    test_errors = np.zeros((num_splits, len(train_percent)))
    #create 2 hidden layer 10 neuron sgd model
    model = tf.keras.models.Sequential()
    model.add(tf.keras.layers.Dense(10,input_dim = len(X0[0,:]),activation='relu'))
    model.add(tf.keras.layers.Dense(10,activation='relu'))
    model.add(tf.keras.layers.Dense(1,activation = "sigmoid"))
    model.compile(loss = 'binary_crossentropy', optimizer = 'SGD',metrics = ['accuracy'])

    split_percent=80
    for i in range(num_splits):
        #create 80-20 split
        Xtrain_i, ytrain_i, Xtest_i, ytest_i = train_test_split(X0, y0, X1, y1,split_percent)
        # try for differnt train set sizes
        for j, percent in enumerate(train_percent):
            #create percent training data
            Xtrain_ij, ytrain_ij = train_data(Xtrain_i, ytrain_i, percent)
            #train model
            model.fit(Xtrain_ij,ytrain_ij,epochs=2,batch_size=20,verbose=0)
            #test model
            loss, acc = model.evaluate(Xtest_i,ytest_i)
            #store results
            test_errors[i][j] = 1-acc

    test2 = np.zeros((num_splits,len(train_percent)))
    #create 2 hidden layer 10 neuron Adam model
    model2= tf.keras.models.Sequential()
    model2.add(tf.keras.layers.Dense(10,input_dim = len(X0[0,:]),activation='relu'))
    model2.add(tf.keras.layers.Dense(10,activation='relu'))
    model2.add(tf.keras.layers.Dense(1,activation = "sigmoid"))
    model2.compile(loss = 'binary_crossentropy', optimizer = 'Adam',metrics = ['accuracy'])
    for i in range(num_splits):
        #create 80-20 split
        Xtrain_i, ytrain_i, Xtest_i, ytest_i = train_test_split(X0, y0, X1, y1,split_percent)
        # try for differnt train set sizes
        for j, percent in enumerate(train_percent):
            #create training percent data
            Xtrain_ij, ytrain_ij = train_data(Xtrain_i, ytrain_i, percent)
            #train
            model2.fit(Xtrain_ij,ytrain_ij,epochs=2,batch_size=20,verbose=0)
            #test
            loss, acc = model.evaluate(Xtest_i,ytest_i)
            #store results
            test2[i][j] = 1-acc
    return test_errors,test2


def logistic(filename, num_splits, train_percent):
```

```
    #parse
    X0, y0, X1, y1 = read_data(filename)

    test_errors = np.zeros((num_splits, len(train_percent)))

    #Create logistic regression SGD model
    model = tf.keras.models.Sequential()
    model.add(tf.keras.layers.Dense(1,input_dim = len(X0[0,:]),activation='sigmoid'))
    model.compile(loss = 'binary_crossentropy', optimizer = 'SGD',metrics = ['accuracy'])

    split_percent=80
    for i in range(num_splits):
        #create 80-20 Split
        Xtrain_i, ytrain_i, Xtest_i, ytest_i = train_test_split(X0, y0, X1, y1,split_percent)
        # try for different train set sizes
        for j, percent in enumerate(train_percent):
            #create percent training data
            Xtrain_ij, ytrain_ij = train_data(Xtrain_i, ytrain_i, percent)
            #train model
            model.fit(Xtrain_ij,ytrain_ij,epochs=2,batch_size=20,verbose=0)
            #test model
            loss, acc = model.evaluate(Xtest_i,ytest_i)
            #store results
            test_errors[i][j] = 1-acc

    test2 = np.zeros((num_splits,len(train_percent)))

    #Create logistic regression Adam model
    model2= tf.keras.models.Sequential()
    model2.add(tf.keras.layers.Dense(1,input_dim = len(X0[0,:]),activation='sigmoid'))
    model2.compile(loss = 'binary_crossentropy', optimizer = 'Adam',metrics = ['accuracy'])
    for i in range(num_splits):
        #create 80-20 split
        Xtrain_i, ytrain_i, Xtest_i, ytest_i = train_test_split(X0, y0, X1, y1,split_percent)
        # try for differnt train set sizes
        for j, percent in enumerate(train_percent):
            #create percent training data
            Xtrain_ij, ytrain_ij = train_data(Xtrain_i, ytrain_i, percent)
            #train model
            model2.fit(Xtrain_ij,ytrain_ij,epochs=2,batch_size=20,verbose=0)
            #test model
            loss, acc = model.evaluate(Xtest_i,ytest_i)
            #store results
            test2[i][j] = 1-acc
    return test_errors,test2

filename='spambase.csv'
num_splits=100
train_percent = [ 10,  20, 30]

#Comment Block For Running different moels

# sgd,adam = logistic(filename,num_splits,train_percent)
# sgd,adam = one_10_relu(filename,num_splits,train_percent)
# sgd,adam = one_10_tanh(filename,num_splits,train_percent)
```

```
# sgd,adam = one_30_relu(filename,num_splits,train_percent)
sgd,adam = two_10_relu(filename,num_splits,train_percent)


#plotting
sgd_mean = np.mean(sgd, axis=0)
adam_mean = np.mean(adam,axis=0)
sgd_stdev = np.std(sgd,axis=0)
adam_stdev =np.std(adam,axis=0)
plt.errorbar(train_percent, sgd_mean, yerr=sgd_stdev, fmt='ro--',capsize=10, elinewidth=2, label="SGD Error")
plt.errorbar(train_percent, adam_mean, yerr=adam_stdev, fmt='bo--', capsize=10, elinewidth=2, label="Adam Error"
)
plt.legend()

plt.xlabel('training %', fontsize=24)
plt.ylabel('test error rate', fontsize=24)
plt.title('Two Hidden, 10 Neuron, Relu: SGD vs. Adam', fontsize=24)
plt.show()
```