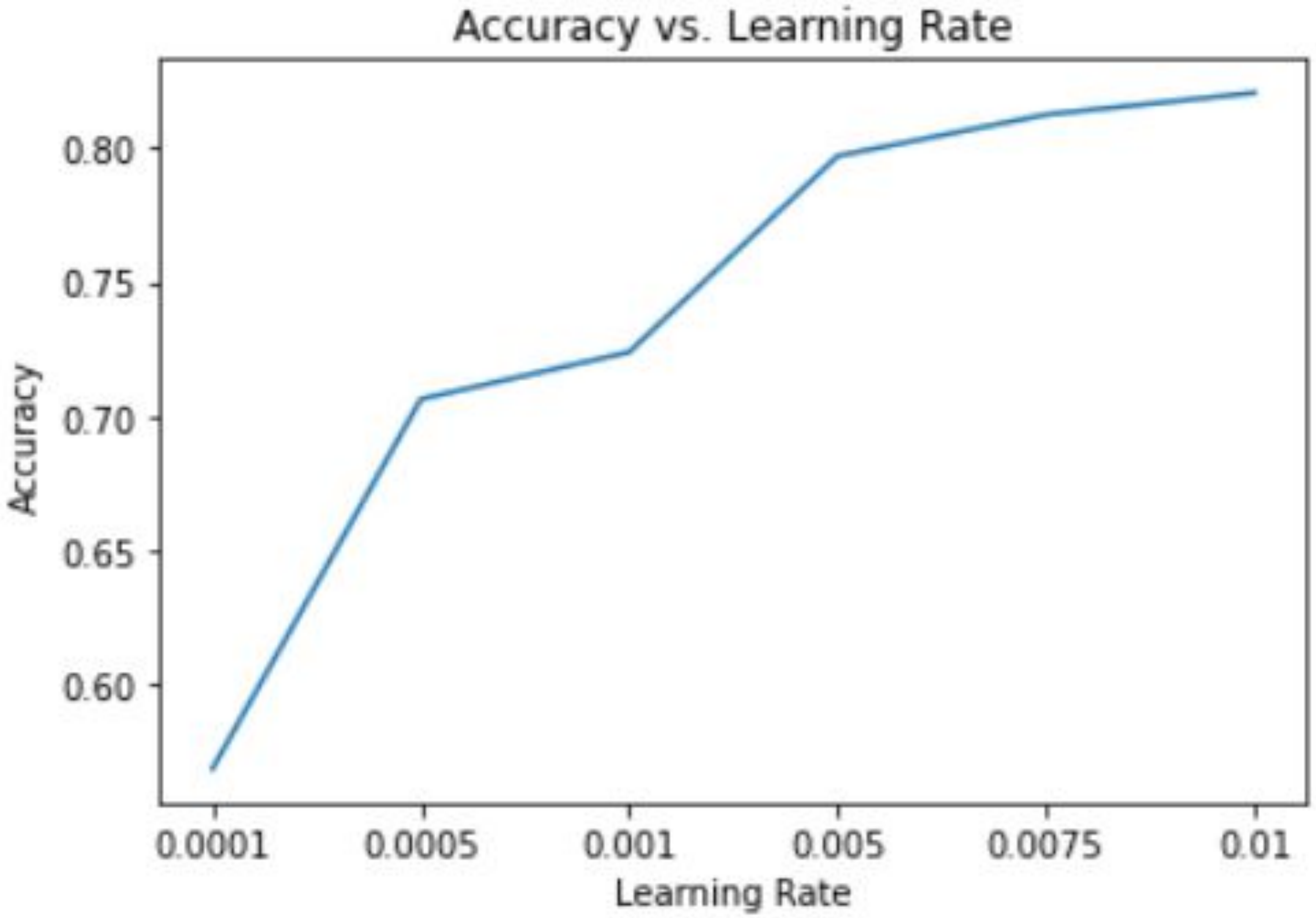
Cory Sweet

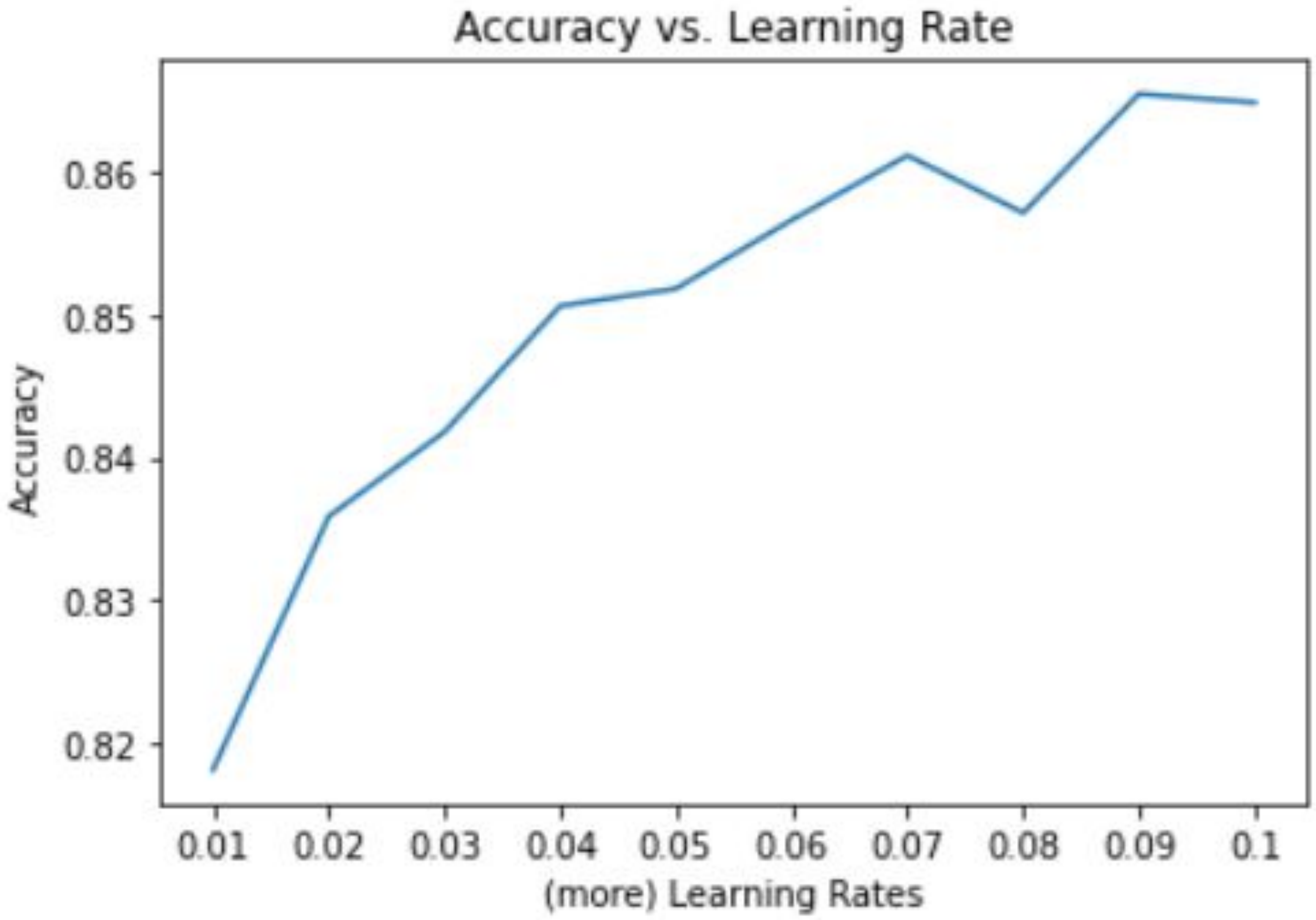
Math 251
Homework 7

Q1



I saw that the accuracy increased with learning rate, so I tried more larger learning rates

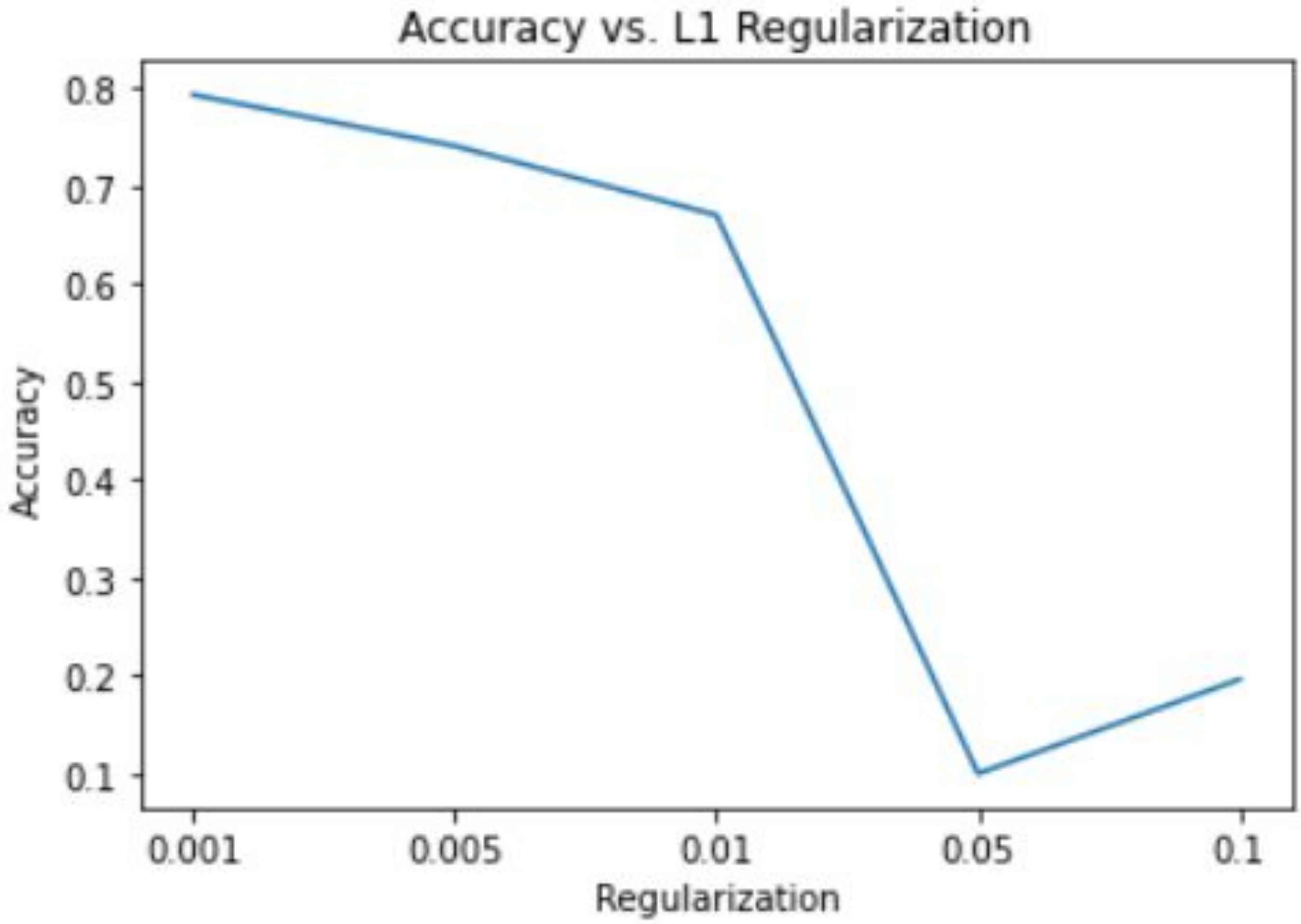
Q1 (cont.)



A learning rate of 0.09 worked best for my network and had an accuracy of: 0.8654. It took roughly 33 seconds, and each step within each epoch took 2ms.

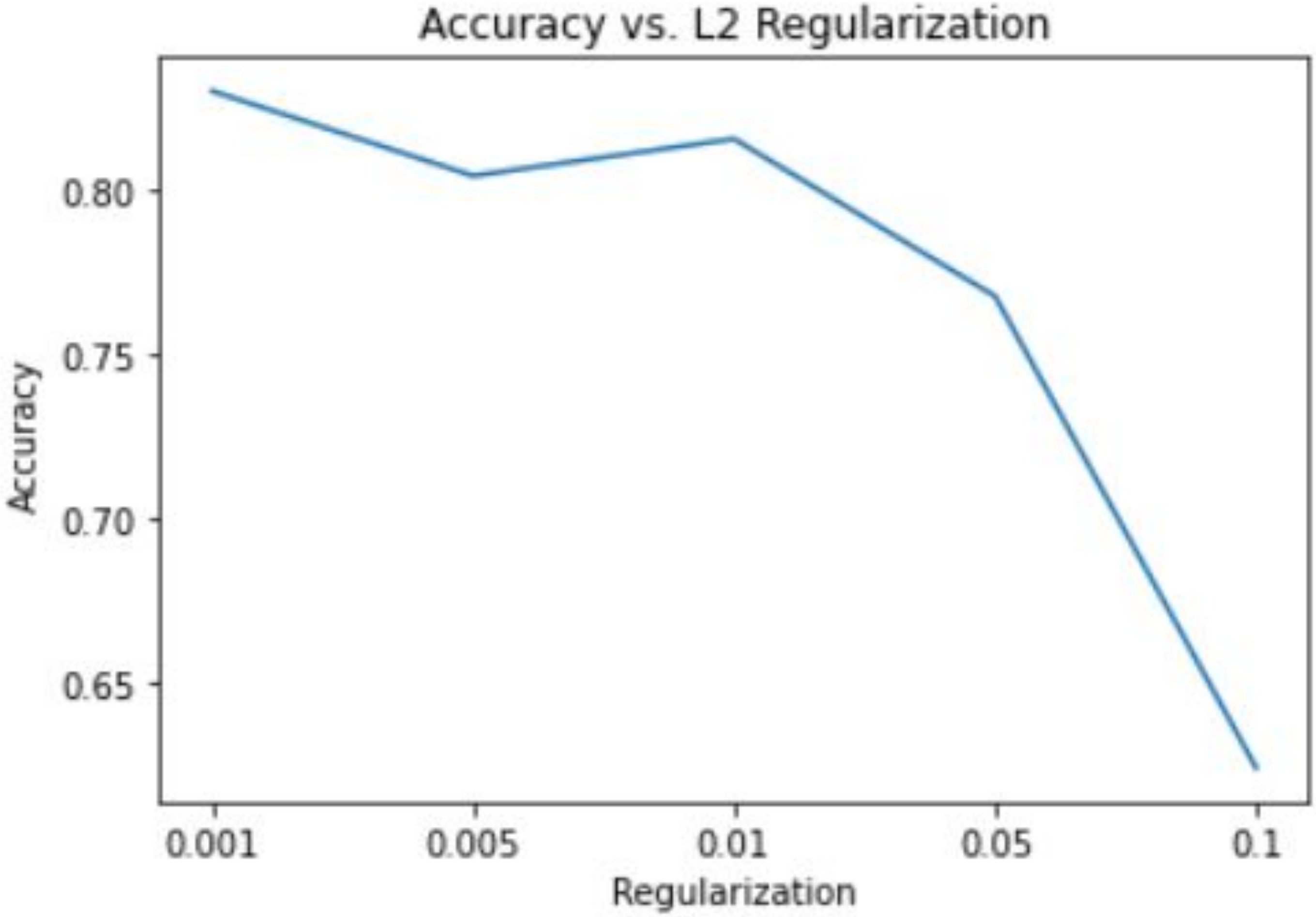
Q2

For this question, I tried L1 regularization, L2 regularization, and both L1 and L2 regularization at the same time. I did this with my best learning rate from part 1: 0.09



The best accuracy of 0.7937 was attained when L1 = 0.001

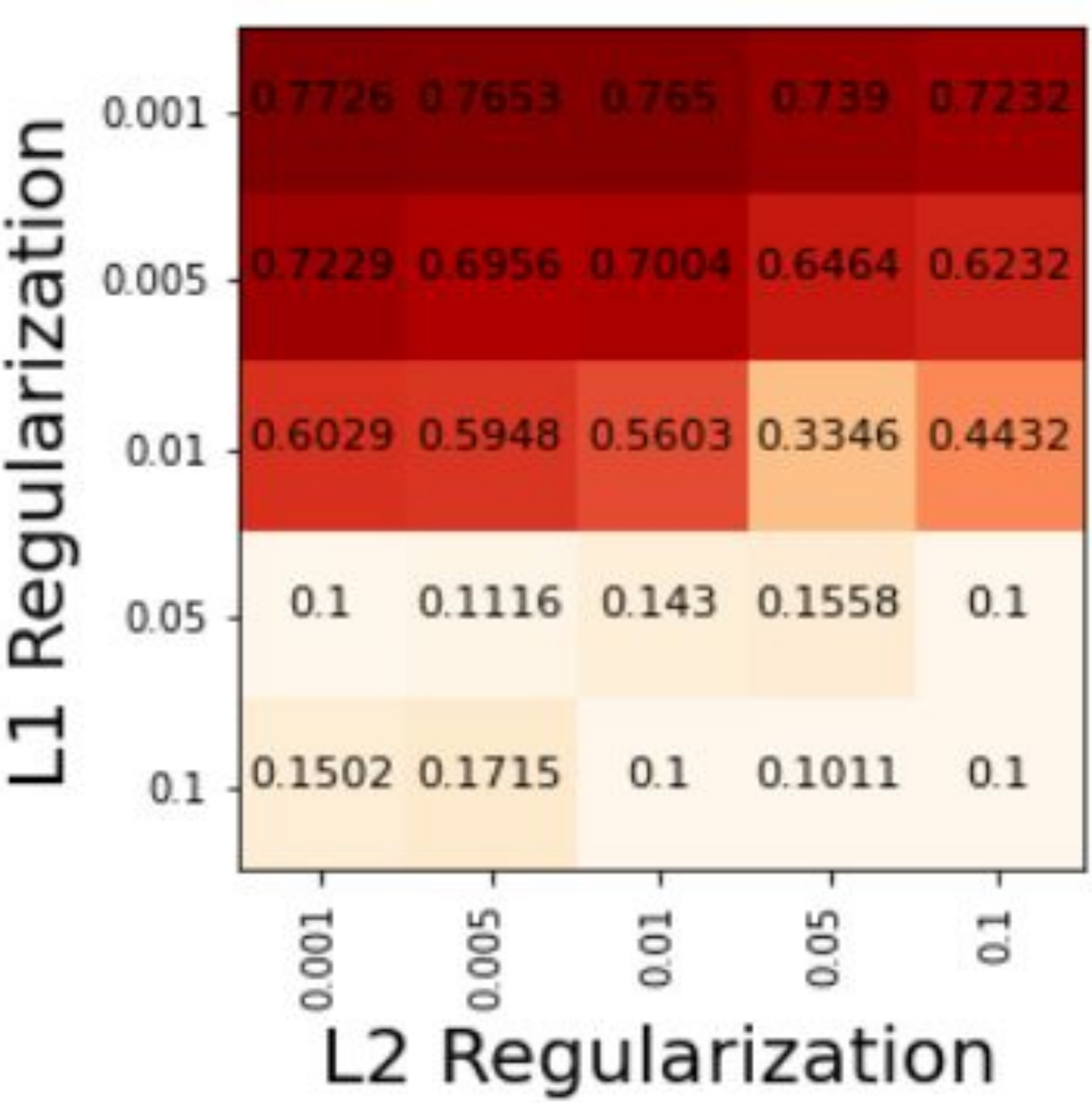
Q2 (cont.)



The best accuracy of 0.8295 was attained when the L2 regularization was set to 0.001

Q2 (cont.)

I also decided to try the l1-l2 regularization in keras. I used the same values for regularization as before, for a total of 25 different models.

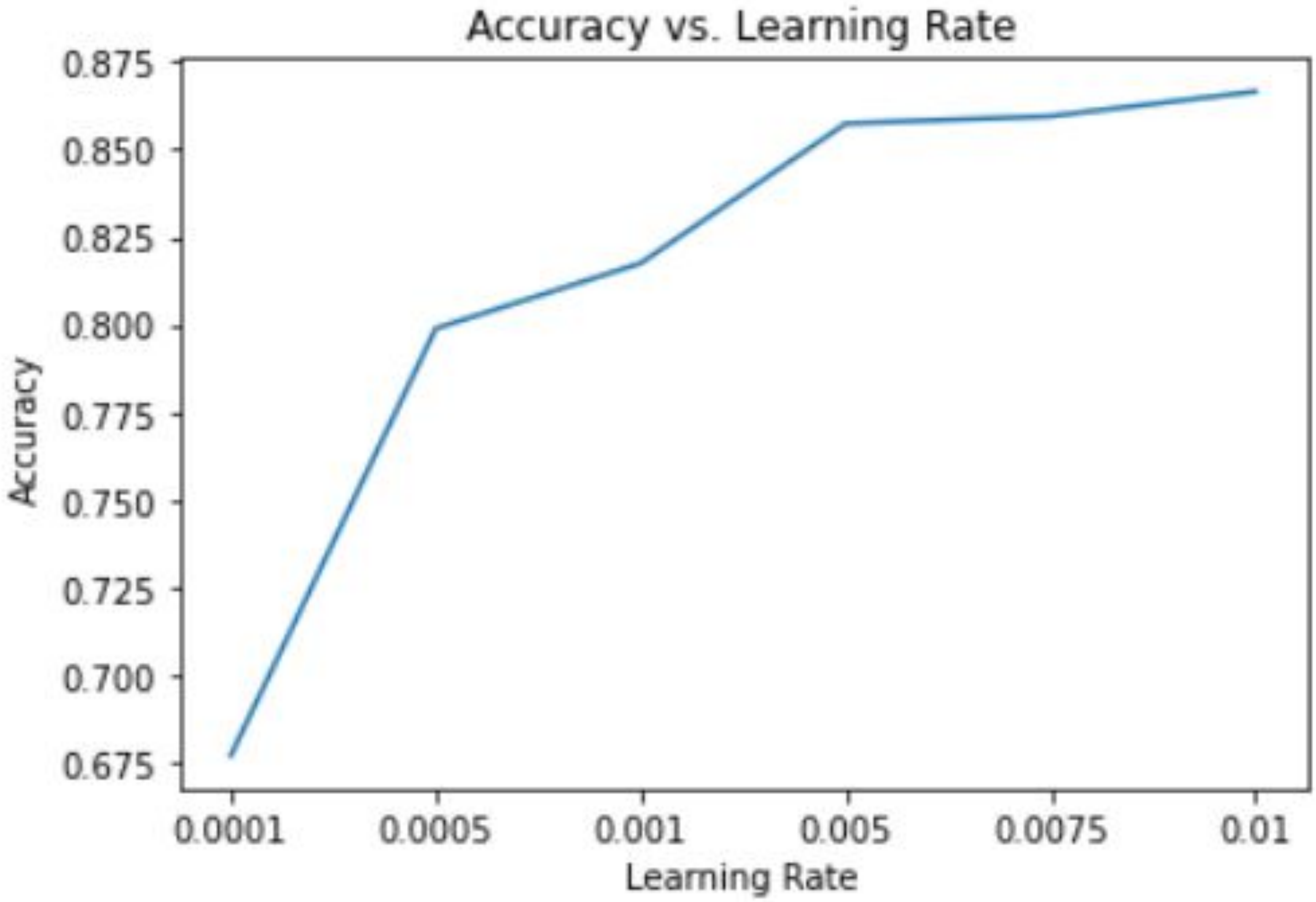


	0.001	0.005	0.01	0.05	0.1
0.001	0.7726	0.7653	0.765	0.739	0.7232
0.005	0.7229	0.6956	0.7004	0.6464	0.6232
0.01	0.6029	0.5948	0.5603	0.3346	0.4432
0.05	0.1	0.1116	0.143	0.1558	0.1
0.1	0.1502	0.1715	0.1	0.1011	0.1

The smaller the regularization terms for both of them together, the better. However, this accuracy (and all of my accuracies in question 2) are worse than my models without regularization.

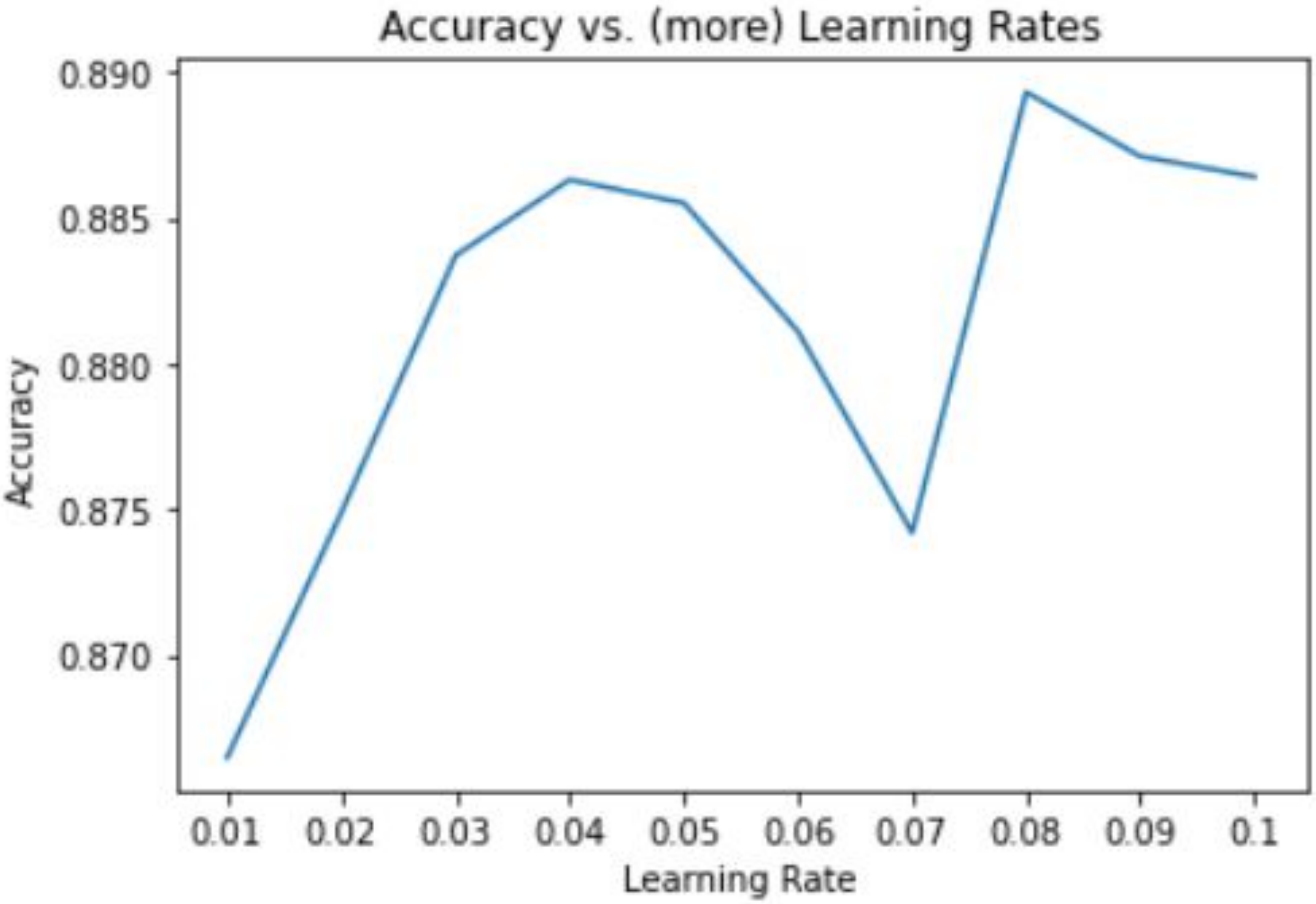
Q3

For question 3, I added a second dense hidden layer with 100 nodes, I changed the activation on both hidden layers to ‘relu’, and I increased the epochs to 20. I also tuned the learning rate similar to question 1. I did not use any regularization.



I saw better results with higher learning rates, so I investigated further

Q3 (cont.)



My best error rate was 0.8893 which happened when the learning rate was 0.08.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras
from sklearn.metrics import accuracy_score
import sklearn.decomposition
import tensorflow as tf
import time

fashion_mnist = keras.datasets.fashion_mnist;
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data();
"""
0       T-shirt/top
1       Trouser
2       Pullover
3       Dress
4       Coat
5       Sandal
6       Shirt
7       Sneaker
8       Bag
9       Ankle boot
"""

label_names=['T-shirt/top','Trouser','Pullover','Dress','Coat','Sandal',
        'Shirt','Sneaker','Bag','Ankle boot']

train_images = train_images/255
test_images = test_images/255

"""
X_train = np.zeros([60000,784])
for i in range(60000):
    img=train_images[i,:,:]
    X_train[i,:] = img.reshape([784])

X_test = np.zeros([10000,784])
for i in range(10000):
    img=test_images[i,:,:]
    X_test[i,:] = img.reshape([784])

X_all = np.vstack((X_train,X_test))
labels = np.hstack((train_labels,test_labels))
"""

#LR_list=[.0001,.0005,.001,.005,.0075,.01]
LR_list = [.01,.02,.03,.04,.05,.06,.07,.08,.09,.1]
accs=[]
```

```
times=[]
for LR in LR_list:
   start=timeit.timeit()
   model = tf.keras.Sequential([
            tf.keras.layers.Flatten(input_shape=(28,28)),
            tf.keras.layers.Dense(100, activation ='sigmoid'),
            tf.keras.layers.Dense(10,activation='softmax')
   ])

   model.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=LR),
            loss='SparseCategoricalCrossentropy',
            metrics=['accuracy'])

   model.fit(train_images, train_labels,batch_size=50,epochs=15)
   probs = model.predict(test_images)
   preds=np.argmax(probs,axis=1)
   end=timeit.timeit()
   #print(LR)
   #print(accuracy_score(y_true = test_labels,y_pred=preds))
   #print(end-start)
   accs.append(accuracy_score(y_true = test_labels,y_pred=preds))
   times.append(end-start)

print(LR_list)
print(accs)
print(times)

#Results: with softmax for final layer
LR_list = [0.0001, 0.0005, 0.001, 0.005, 0.0075, 0.01]
accuracy_list = [0.5689, 0.7065, 0.724, 0.797, 0.8124, 0.8205]
times = [-0.005110244000206876, -1.4107000424701255e-05, 0.002290139001161151,
3.0379997042473406e-06, 0.002505352000298444, 0.001273362000574707]

plt.plot(range(6),accuracy_list)
plt.xticks(range(6),labels=LR_list)
plt.xlabel('Learning Rate')
plt.ylabel('Accuracy')
plt.title('Accuracy vs. Learning Rate');

LR_list = [0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1]
acc_list = [0.8182, 0.8359, 0.8418, 0.8506, 0.8518, 0.8566, 0.8611, 0.8571, 0.8654, 0.8648]

plt.plot(range(10),acc_list)
plt.xticks(range(10),labels=LR_list)
plt.xlabel('(more) Learning Rates')
plt.ylabel('Accuracy')
plt.title('Accuracy vs. Learning Rate');
```

```
#just to get the time
import time
start=time.time()
model = tf.keras.Sequential([
        tf.keras.layers.Flatten(input_shape=(28,28)),
        tf.keras.layers.Dense(100, activation ='sigmoid'),
        tf.keras.layers.Dense(10,activation='softmax')
])

model.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=.09),
        loss='SparseCategoricalCrossentropy',
        metrics=['accuracy'])

model.fit(train_images, train_labels,batch_size=50,epochs=15)
probs = model.predict(test_images)
preds=np.argmax(probs,axis=1)
end=time.time()
print(end-start)

############################################
#2

L1_list = [.001,.005,.01,.05,.1]
accs=[]
LR=.09
for L1 in L1_list:
  model = tf.keras.Sequential([
          tf.keras.layers.Flatten(input_shape=(28,28)),
          tf.keras.layers.Dense(100, activation ='sigmoid',
            kernel_regularizer=tf.keras.regularizers.l1(L1)),
          tf.keras.layers.Dense(10,activation='softmax')
  ])

  model.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=LR),
          loss='SparseCategoricalCrossentropy',
          metrics=['accuracy'])

  model.fit(train_images, train_labels,batch_size=50,epochs=15)
  probs = model.predict(test_images)
  preds=np.argmax(probs,axis=1)

  #print(LR)
  print(accuracy_score(y_true = test_labels,y_pred=preds))
  accs.append(accuracy_score(y_true = test_labels,y_pred=preds))

print(L1_list)
print(accs)
```

```
#L1 results
L1_list = [0.001, 0.005, 0.01, 0.05, 0.1]
accuracy_list = [0.7937, 0.7414, 0.6707, 0.1, 0.1963]

plt.plot(range(5),accuracy_list)
plt.xticks(range(5),labels=L1_list)
plt.xlabel('Regularization')
plt.ylabel('Accuracy')
plt.title('Accuracy vs. L1 Regularization');

L2_list = [.001,.005,.01,.05,.1]
accs=[]
LR=.09
for L2 in L2_list:
   model = tf.keras.Sequential([
            tf.keras.layers.Flatten(input_shape=(28,28)),
            tf.keras.layers.Dense(100, activation ='sigmoid',
               kernel_regularizer=tf.keras.regularizers.l2(L2)),
            tf.keras.layers.Dense(10,activation='softmax')
   ])

   model.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=LR),
            loss='SparseCategoricalCrossentropy',
            metrics=['accuracy'])

   model.fit(train_images, train_labels,batch_size=50,epochs=15)
   probs = model.predict(test_images)
   preds=np.argmax(probs,axis=1)

   #print(LR)
   print(accuracy_score(y_true = test_labels,y_pred=preds))
   accs.append(accuracy_score(y_true = test_labels,y_pred=preds))

print(L1_list)
print(accs)

L2_list = [0.001, 0.005, 0.01, 0.05, 0.1]
acc_list = [0.8295, 0.8038, 0.815, 0.7673, 0.6244]

plt.plot(range(5),acc_list)
plt.xticks(range(5),labels=L2_list)
plt.xlabel('Regularization')
plt.ylabel('Accuracy')
plt.title('Accuracy vs. L2 Regularization');

L1_list = [.001,.005,.01,.05,.1]

L2_list = [.001,.005,.01,.05,.1]
```

```
accs=np.zeros([5,5])
LR=.01
for a,L1 in enumerate(L1_list):
   for b,L2 in enumerate(L2_list):
      model = tf.keras.Sequential([
               tf.keras.layers.Flatten(input_shape=(28,28)),
               tf.keras.layers.Dense(100, activation ='sigmoid',
                  kernel_regularizer=tf.keras.regularizers.l1_l2(L1,L2)),
               tf.keras.layers.Dense(10,activation='softmax')
      ])

      model.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=LR),
               loss='SparseCategoricalCrossentropy',
               metrics=['accuracy'])

      model.fit(train_images, train_labels,batch_size=50,epochs=15)
      probs = model.predict(test_images)
      preds=np.argmax(probs,axis=1)

      accs[a,b] = accuracy_score(y_true = test_labels,y_pred=preds)

print(accs)
#rows are L1, columns are L2

fig,ax = plt.subplots()
ax.imshow(accs,cmap='OrRd')
ax.set_xticks(np.arange(5))
ax.set_yticks(np.arange(5))
ax.set_xticklabels([.001,.005,.01,.05,.1],rotation = 90)
ax.set_yticklabels([.001,.005,.01,.05,.1])
ax.set_ylim(len(accs)-.5,-.5)
ax.set_xlabel('L2 Regularization',size=20)
ax.set_ylabel('L1 Regularization',size=20)


for i in range(5):
   for j in range(5):
      test = ax.text(j,i,accs[i,j], ha='center', color='k',size=11)

####################################
#3

LR_list=[.0001,.0005,.001,.005,.0075,.01]
accs=[]

for LR in LR_list:
   model = tf.keras.Sequential([
            tf.keras.layers.Flatten(input_shape=(28,28)),
                tf.keras.layers.Dense(100, activation ='relu'),
                tf.keras.layers.Dense(100, activation ='relu'),
                tf.keras.layers.Dense(10,activation='softmax')
    ])

    model.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=LR),
                loss='SparseCategoricalCrossentropy',
                metrics=['accuracy'])

    model.fit(train_images, train_labels,batch_size=50,epochs=20)
    probs = model.predict(test_images)
    preds=np.argmax(probs,axis=1)
    accs.append(accuracy_score(y_true = test_labels,y_pred=preds))

print(LR_list)
print(accs)

LR_list = [0.0001, 0.0005, 0.001, 0.005, 0.0075, 0.01]
acc_list = [0.6773, 0.7989, 0.8175, 0.8573, 0.8594, 0.8664]

plt.plot(range(6),acc_list)
plt.xticks(range(6),labels=LR_list)
plt.xlabel('Learning Rate')
plt.ylabel('Accuracy')
plt.title('Accuracy vs. Learning Rate');

LR_list=[.01,.02,.03,.04,.05,.06,.07,.08,.09,0.1]
accs=[]

for LR in LR_list:
    model = tf.keras.Sequential([
                tf.keras.layers.Flatten(input_shape=(28,28)),
                tf.keras.layers.Dense(100, activation ='relu'),
                tf.keras.layers.Dense(100, activation ='relu'),
                tf.keras.layers.Dense(10,activation='softmax')
    ])

    model.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=LR),
                loss='SparseCategoricalCrossentropy',
                metrics=['accuracy'])

    model.fit(train_images, train_labels,batch_size=50,epochs=20)
    probs = model.predict(test_images)
    preds=np.argmax(probs,axis=1)
    accs.append(accuracy_score(y_true = test_labels,y_pred=preds))

print(LR_list)
print(accs)
```

```
plt.plot(range(10),accs)
plt.xticks(range(10),labels=LR_list)
plt.xlabel('Learning Rate')
plt.ylabel('Accuracy')
plt.title('Accuracy vs. (more) Learning Rates');
```