uOttawa

Faculté de génie
Faculty of Engineering

Applied Machine Learning - Summer 2021

Assignment 5 - MLP

Submitted by:

Name	Email
Youssef Metwally	ymetw027@uOttawa.ca
Dina Ibrahim Morsy	dabde007@uOttawa.ca
Adel El Nabarawy	aelna025@uOttawa.ca

Submitted to:

Dr. Murat SIMSEK, SMIEEE

Q1:

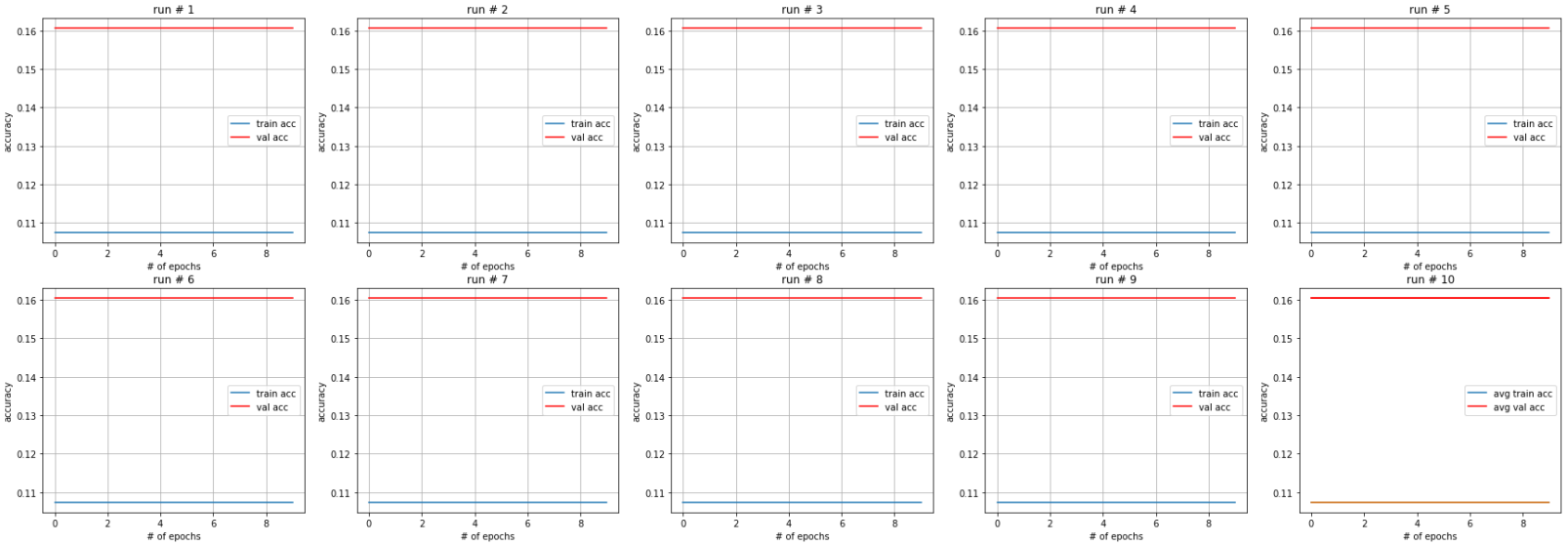
Apply Multi Layer Perceptron (MLP) on the provided dataset by using activation functions listed below. Use the given parameters. Run MLP 10 times for each case, plot runtime vs. accuracy with average line as shown in below.

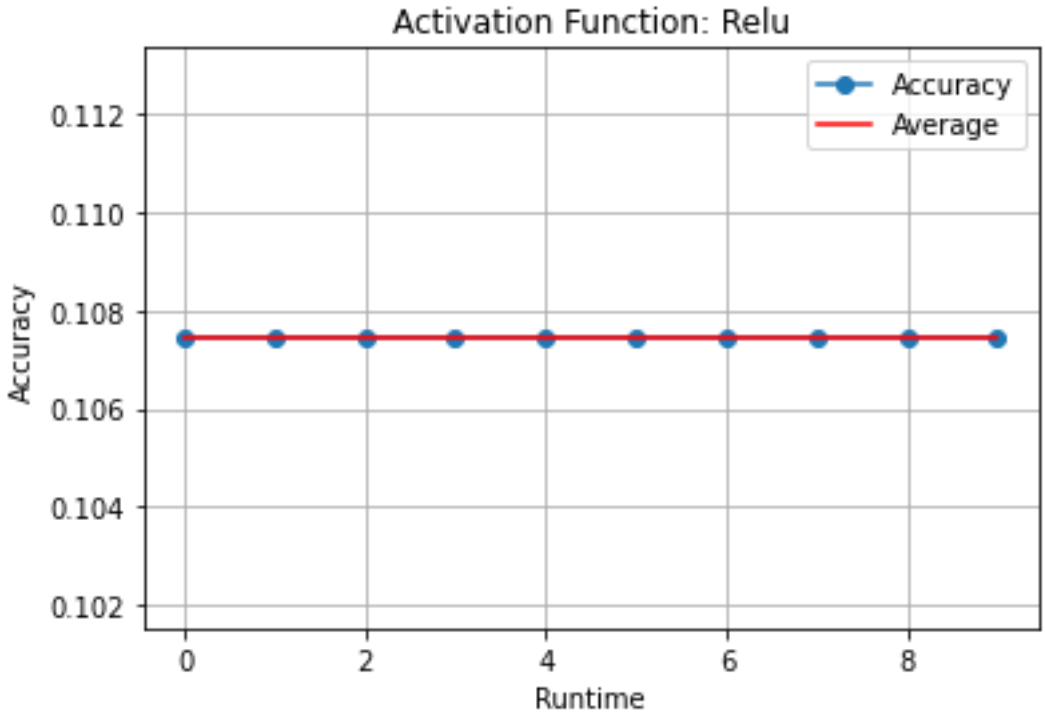
```
models = []
model = keras.models.Sequential([
   Dense(5, activation='relu', input_shape=(5,)),
   Dense(4, activation='relu')
 ])
models.append(model)
model = keras.models.Sequential([
   Dense(5, activation='sigmoid', input_shape=(5,)),
   Dense(4, activation='sigmoid')
 ])
models.append(model)
model = keras.models.Sequential([
   Dense(5, activation='tanh', input_shape=(5,)),
   Dense(4, activation='tanh')
 ])
models.append(model)
print(models[0].summary())

mod-
els[0].compile(loss='sparse_categorical_crossentropy', optimizer=keras.optimizers
.Adam(learning_rate=0.1), metrics=['accuracy'])
mod-
els[1].compile(loss='sparse_categorical_crossentropy', optimizer=keras.optimizers
.Adam(learning_rate=0.1), metrics=['accuracy'])
mod-
els[2].compile(loss='sparse_categorical_crossentropy', optimizer=keras.optimizers
.Adam(learning_rate=0.1), metrics=['accuracy'])

h_total_train=np.empty((0,10))
h_total_val=np.empty((0,10))
total_acc=[]
```

```
number_of_runs=10
for i in range(number_of_runs):
  h = models[0].fit(X_train, y_train, epochs=1000, batch_size=32, validation_spli
t=0.25, verbose=0)
  h_total_train=np.append(h_total_train,np.array([h.history['accuracy']]),axis=0)
  h_total_val=np.append(h_total_val,np.array([h.history['val_accuracy']]),axis=0)
  total_acc.append(models[0].evaluate(X_test,y_test, verbose=0)[1])

for i in range(number_of_runs):
  f=plt.subplot(2,5,i+1)
  f.figure.set_size_inches(30,10)
  plt.plot(h_total_train[i])
  plt.plot(h_total_val[i],'r')
  plt.legend(['train acc', 'val acc'])
  plt.title("run # %d"%(i+1))
  plt.xlabel("# of epochs")
  plt.ylabel("accuracy")
  plt.grid()

h_avg_train=np.average(h_total_train,axis=0)
h_avg_val=np.average(h_total_val,axis=0)
plt.plot(h_avg_train)
plt.plot(h_avg_val, 'r')
plt.legend(['avg train acc', 'avg val acc'])
plt.xlabel("# of epochs")
plt.ylabel("accuracy")
plt.grid()

print('Test accuracy: %.2f %%'%(100*mean(total_acc)))
plt.show()

ten_runs_train=[]
ten_runs_val=[]
for i in range(number_of_runs):
  ten_runs_train.append(h_total_train[i][-1])
```

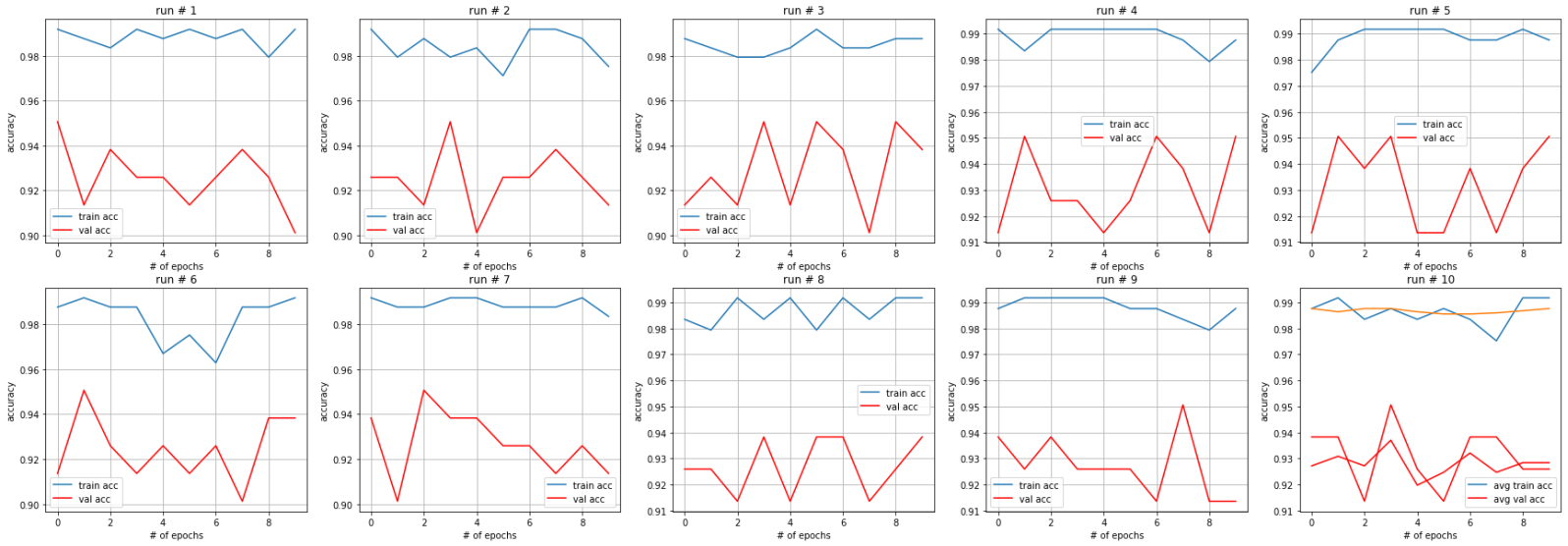
```
  ten_runs_val.append(h_total_val[i][-1])


plt.plot(ten_runs_train, marker='o')
plt.plot(range(0,10),[h_avg_train[-1]]*10,'r')
plt.legend(['Accuracy', 'Average'])
plt.xlabel("Runtime")
plt.ylabel("Accuracy")
plt.title("Activation Function: Relu")
plt.grid()
plt.show()
```

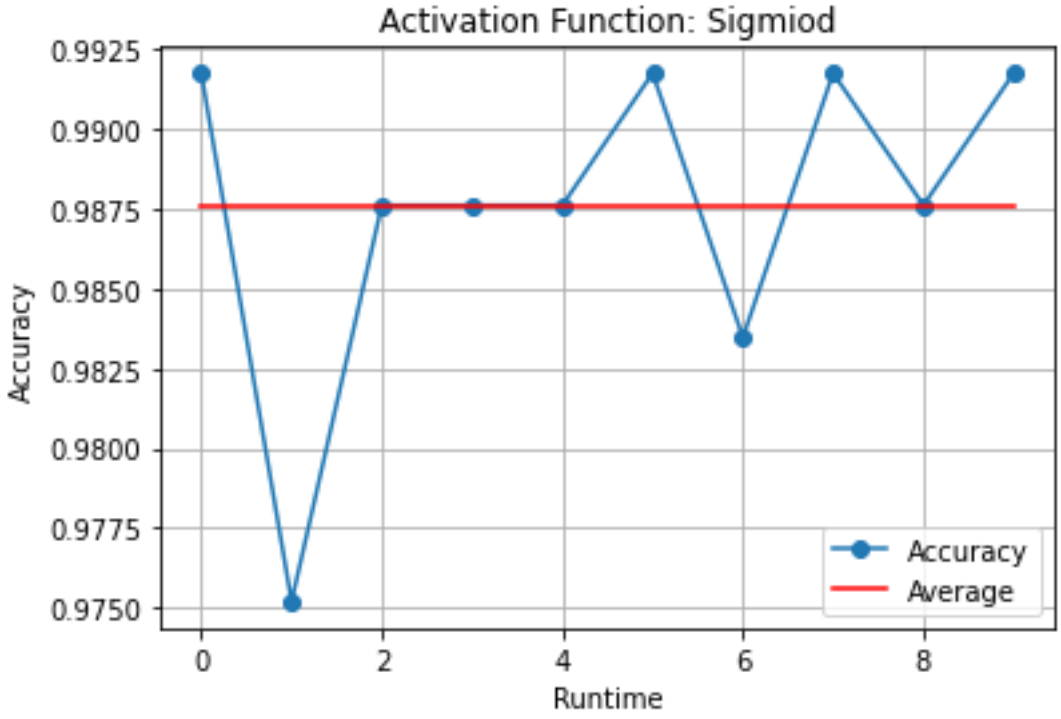
Test accuracy: 13.75 %



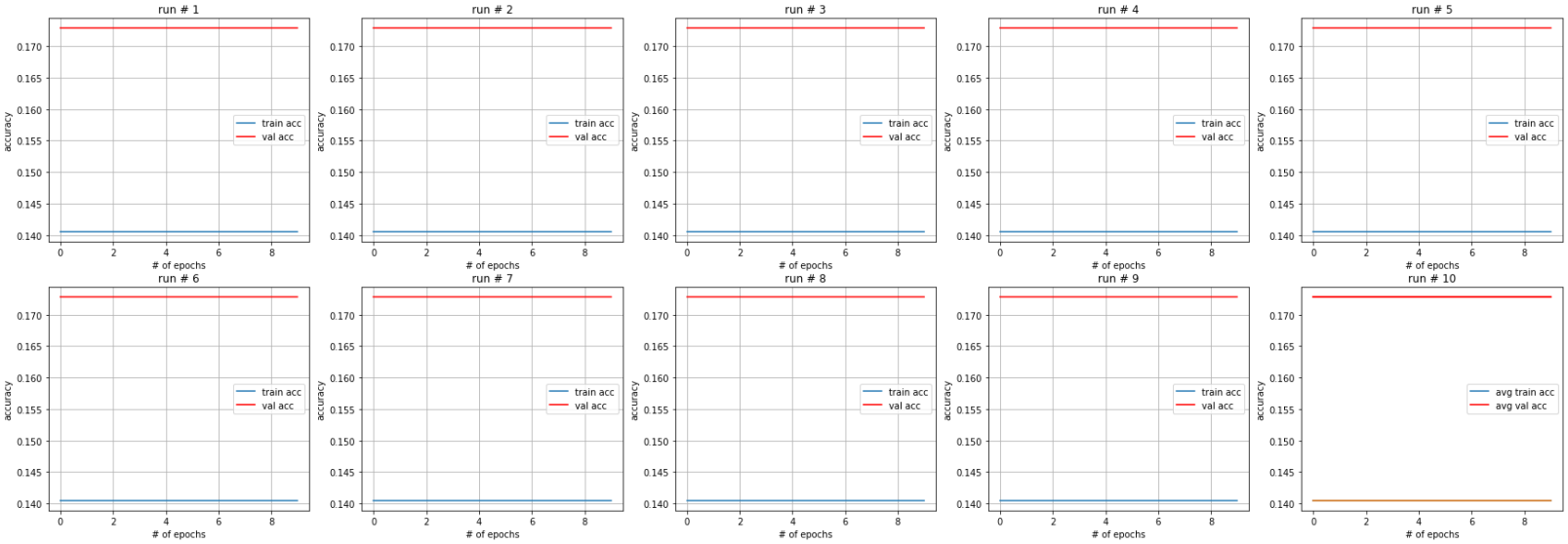


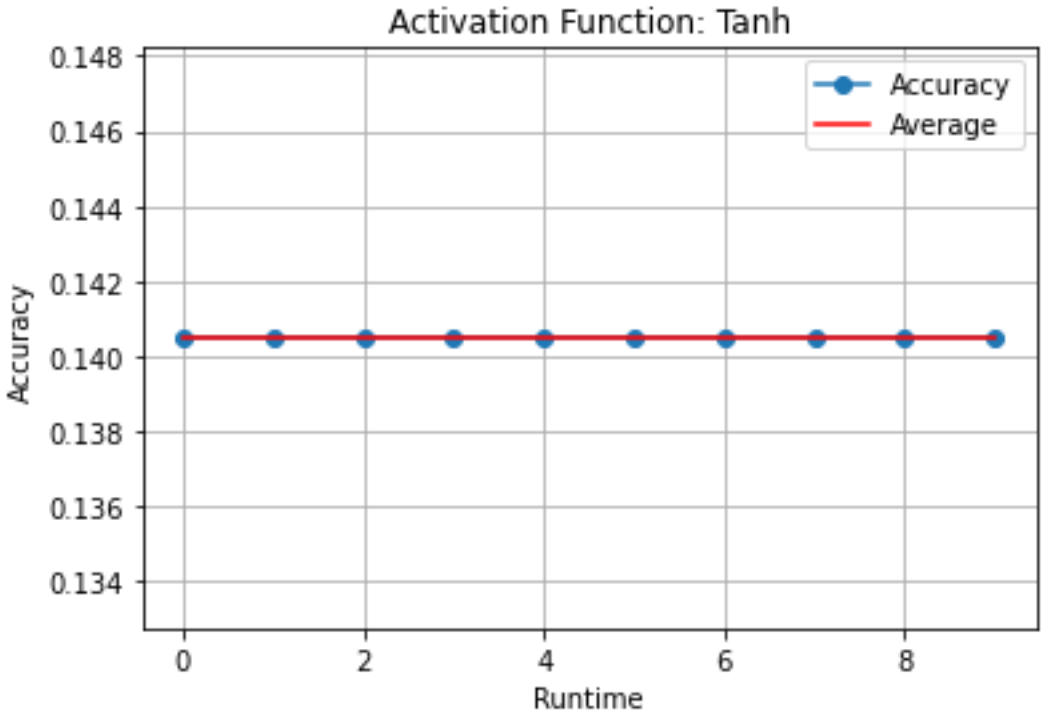
Test accuracy: 95.75 %





Test accuracy: 17.50 %





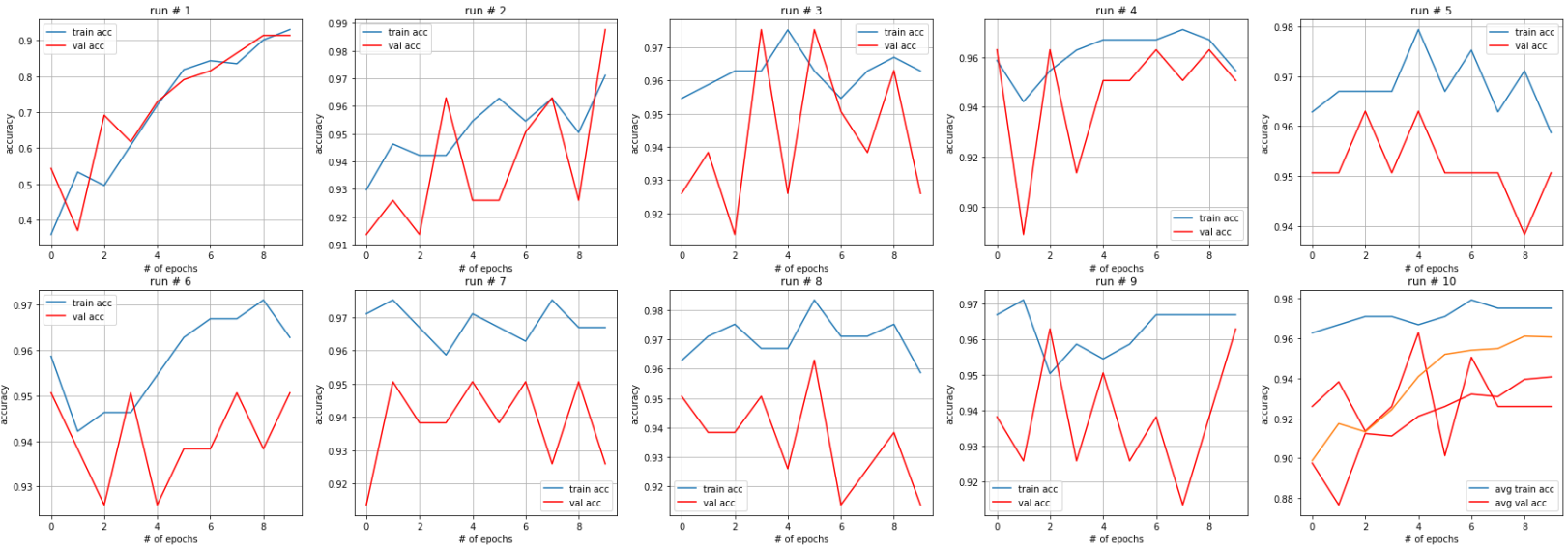
Q2:

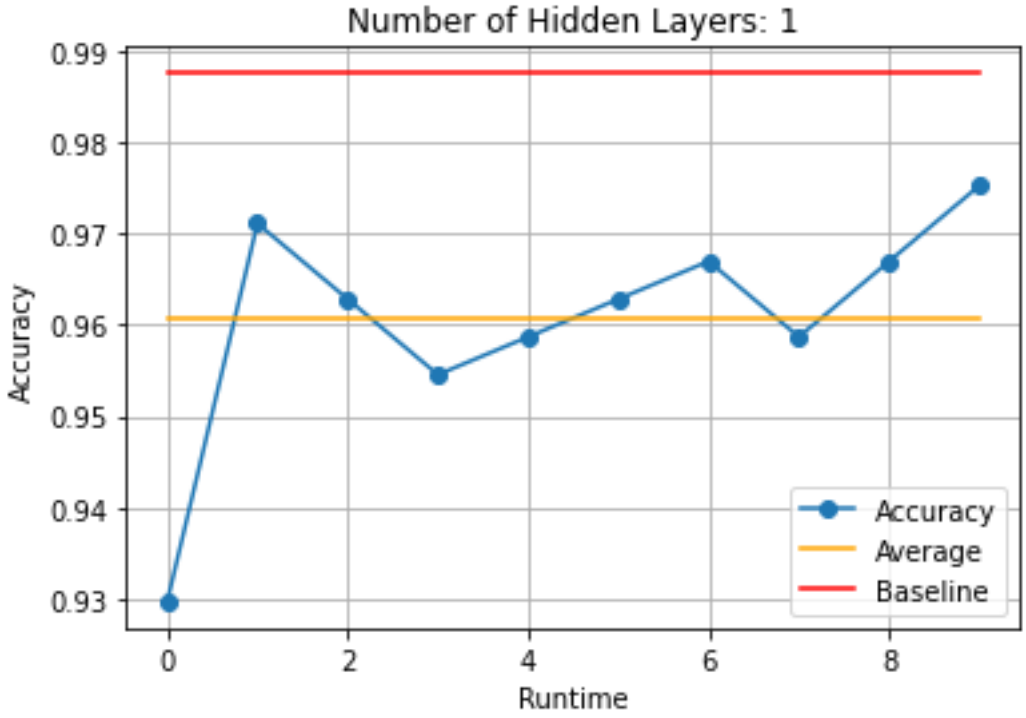
Choose the activation function that provides highest average accuracy value in Q1. Try different number of hidden layers as given below. Run MLP 10 times for each case, plot runtime vs. accuracy with average line. Plot the best average accuracy value from Q1 as baseline performance.

```
# 1 Hidden layers
model_1h = keras.models.Sequential([
   Dense(5, activation='sigmoid', input_shape=(5,)),
   Dense(4, activation='sigmoid')
 ])

mod-
el_1h.compile(loss='sparse_categorical_crossentropy', optimizer=keras.optimizers.
Adam(learning_rate=0.1), metrics=['accuracy'])

base-
line = [0.9875, 0.9875, 0.9875, 0.9875, 0.9875, 0.9875, 0.9875, 0.9875, 0.9875, 0
.9875]

h_total_train=np.empty((0,10))
h_total_val=np.empty((0,10))
total_acc=[]
number_of_runs=10
for i in range(number_of_runs):
  h = model_1h.fit(X_train, y_train, epochs=1000, batch_size=32, validation_split
=0.25, verbose=0)
  h_total_train=np.append(h_total_train,np.array([h.history['accuracy']]),axis=0)
  h_total_val=np.append(h_total_val,np.array([h.history['val_accuracy']]),axis=0)
  total_acc.append(model_1h.evaluate(X_test,y_test, verbose=0)[1])

for i in range(number_of_runs):
  f=plt.subplot(2,5,i+1)
  f.figure.set_size_inches(30,10)
  plt.plot(h_total_train[i])
  plt.plot(h_total_val[i],'r')
```

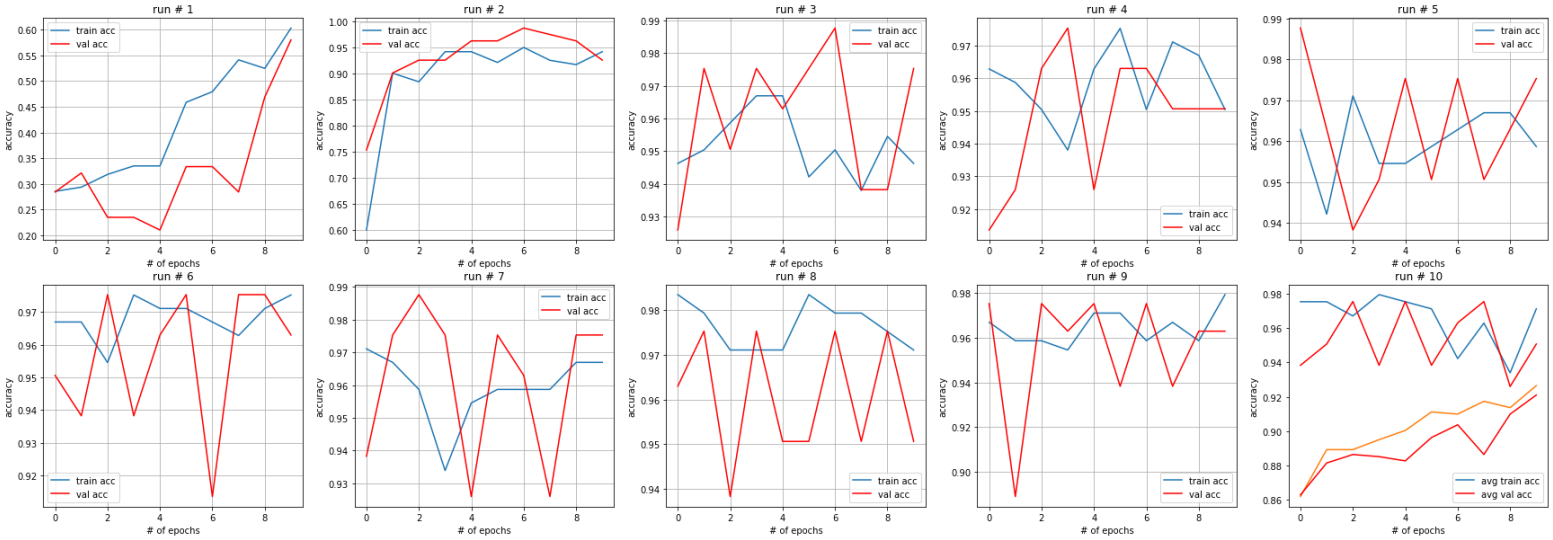
```
  plt.legend(['train acc', 'val acc'])
  plt.title("run # %d"%(i+1))
  plt.xlabel("# of epochs")
  plt.ylabel("accuracy")
  plt.grid()

h_avg_train=np.average(h_total_train,axis=0)
h_avg_val=np.average(h_total_val,axis=0)
plt.plot(h_avg_train)
plt.plot(h_avg_val, 'r')
plt.legend(['avg train acc', 'avg val acc'])
plt.xlabel("# of epochs")
plt.ylabel("accuracy")
plt.grid()
plt.show()

print('Test accuracy: %.2f %%'%(100*mean(total_acc)))

ten_runs_train=[]
ten_runs_val=[]
for i in range(number_of_runs):
  ten_runs_train.append(h_total_train[i][-1])
  ten_runs_val.append(h_total_val[i][-1])

plt.plot(ten_runs_train, marker='o')
plt.plot(range(0,10),[h_avg_train[-1]]*10,color='orange')
plt.plot(range(0,10),baseline,'r')
plt.legend(['Accuracy', 'Average', 'Baseline'])
plt.xlabel("Runtime")
plt.ylabel("Accuracy")
plt.title("Number of Hidden Layers: 1")
plt.grid()
plt.show()
```

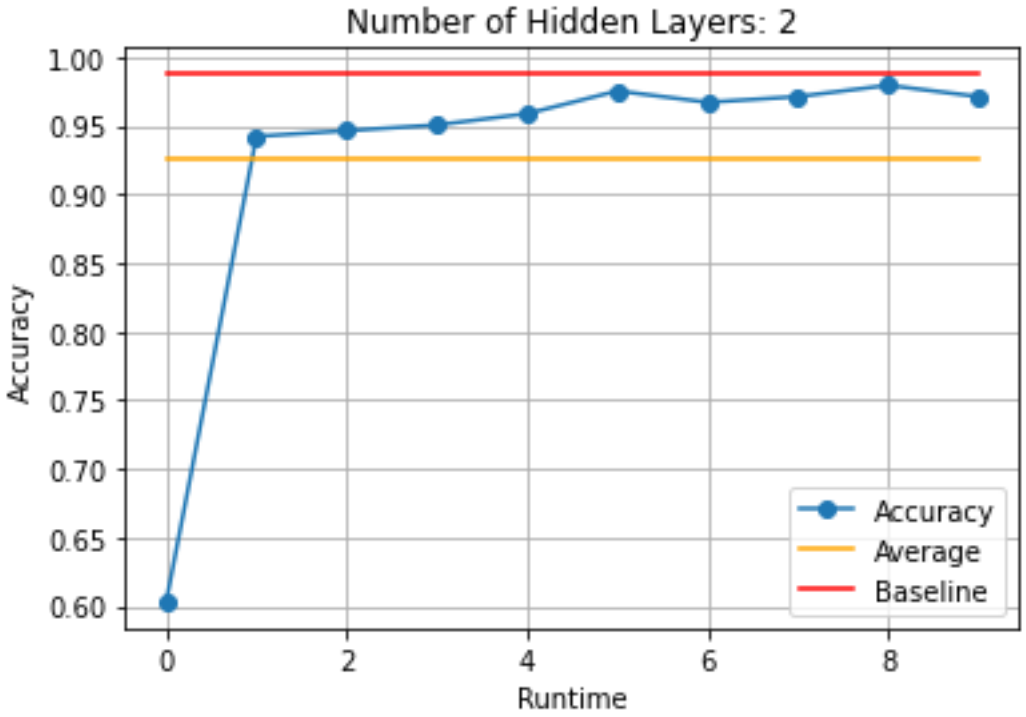
Test accuracy: 96.75 %



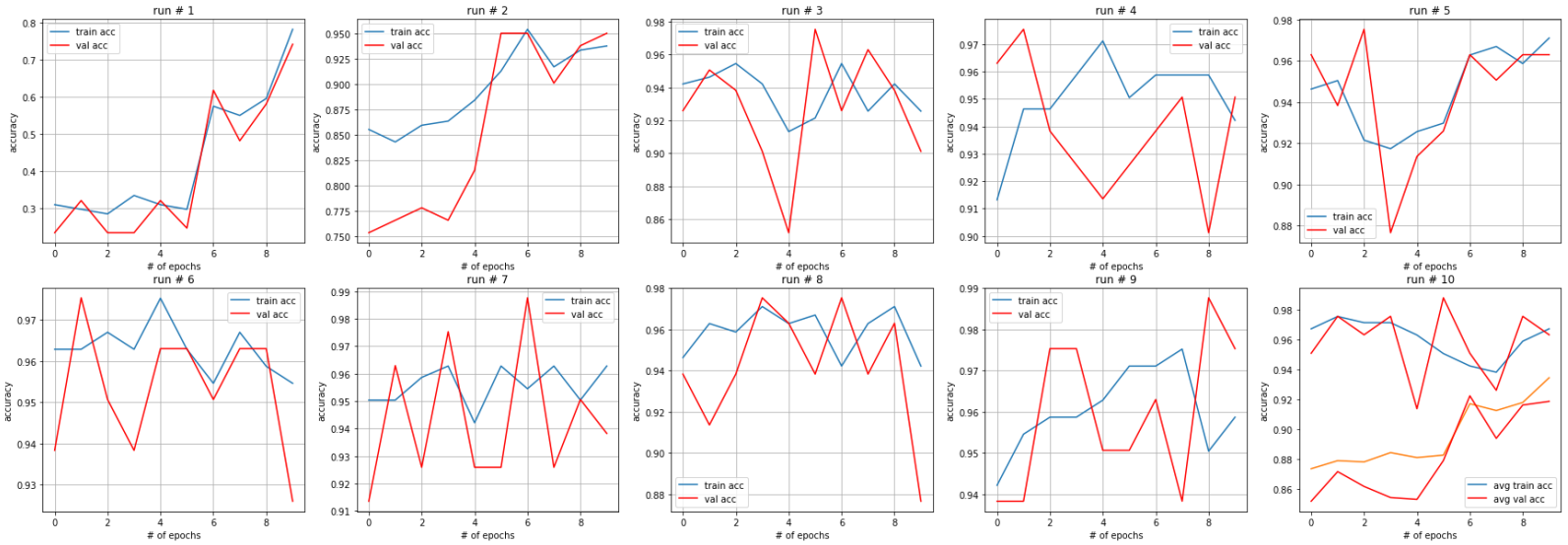


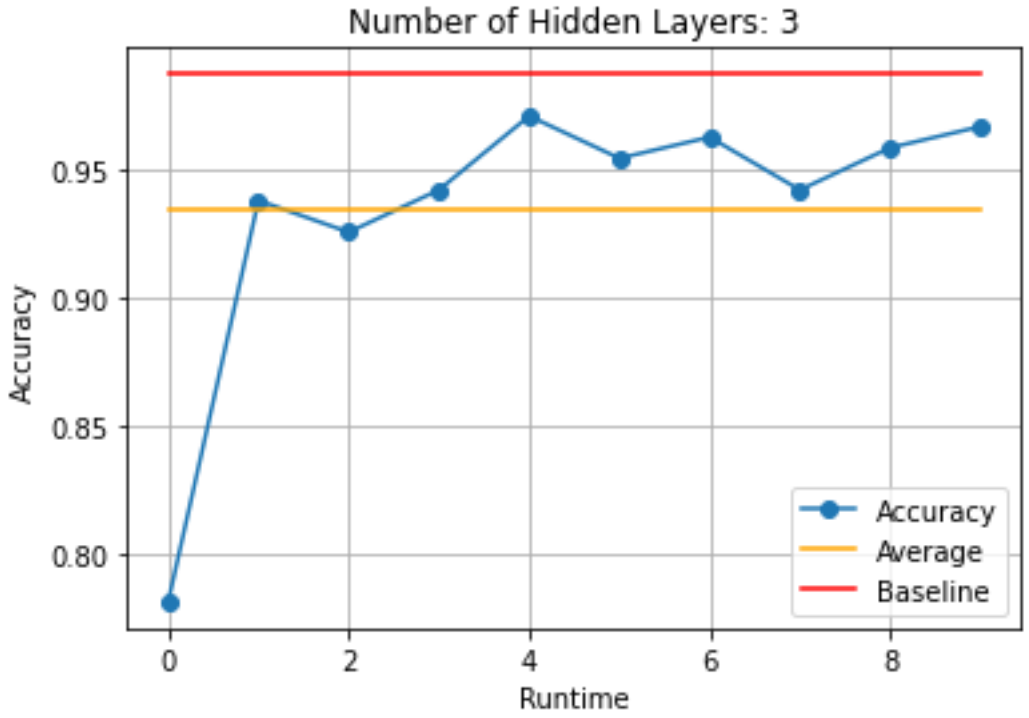
Test accuracy: 92.63 %



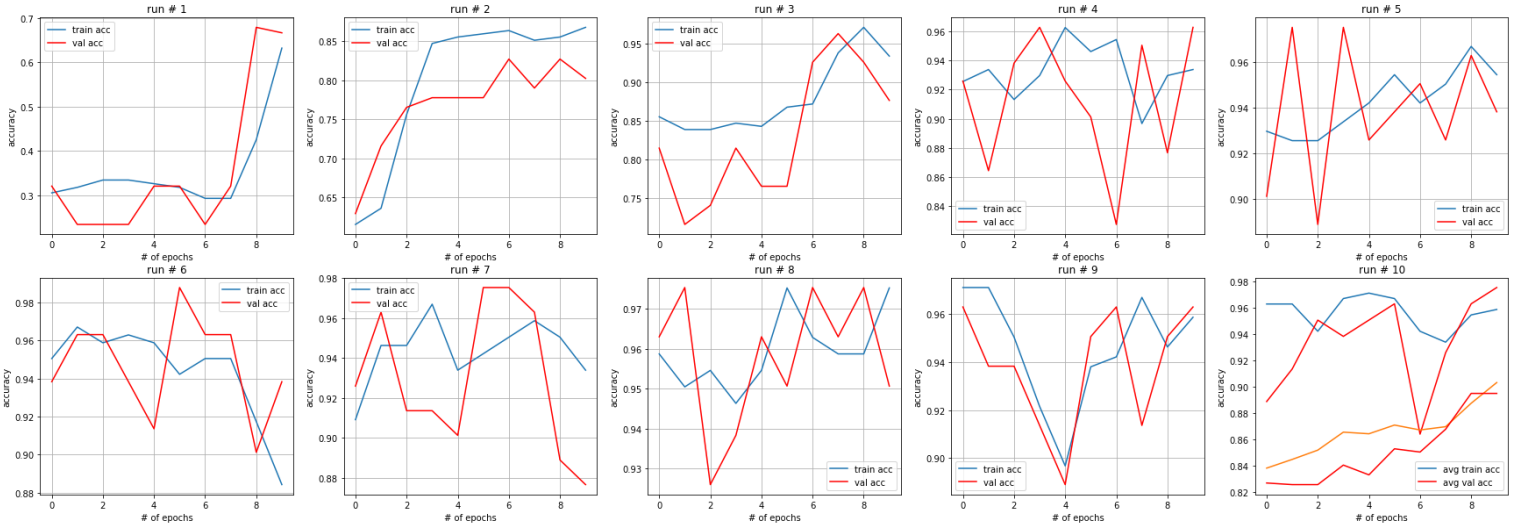


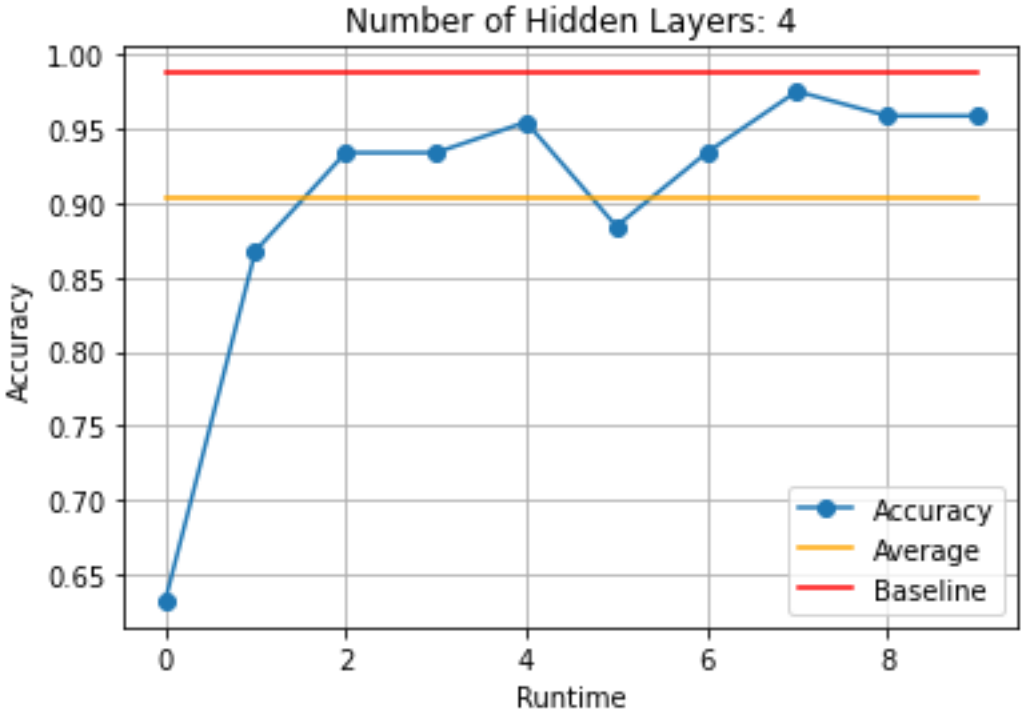
Test accuracy: 94.00 %





Test accuracy: 91.13 %





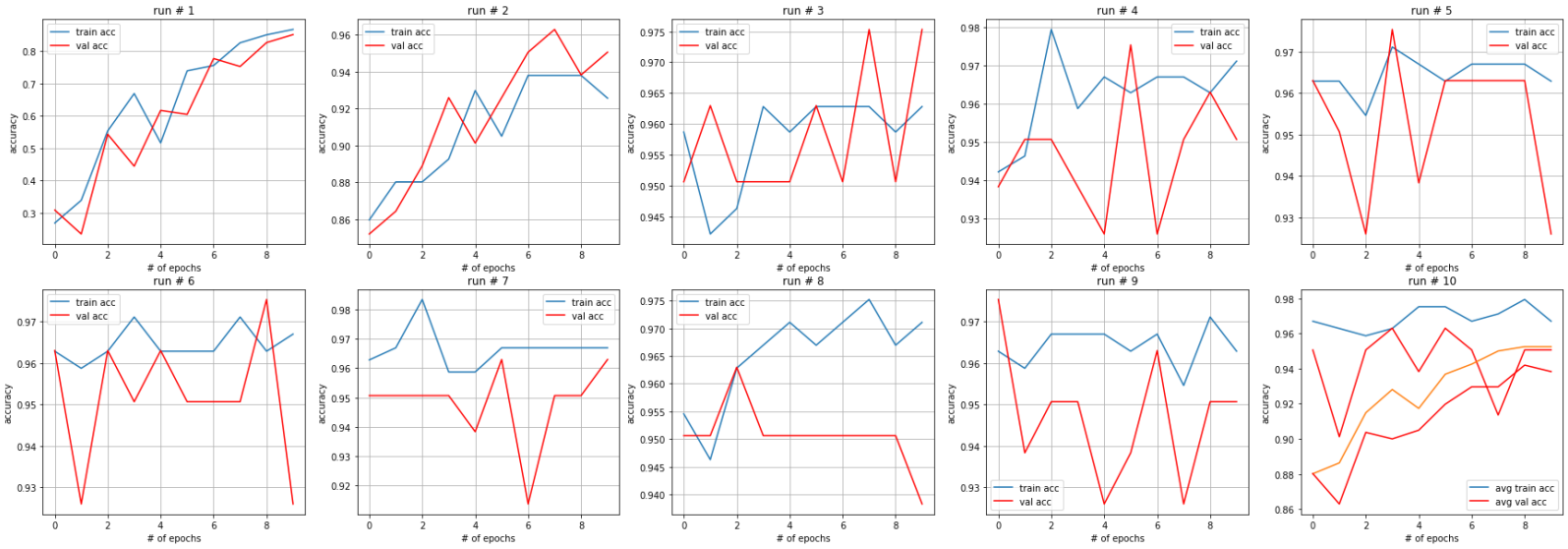
Q3:

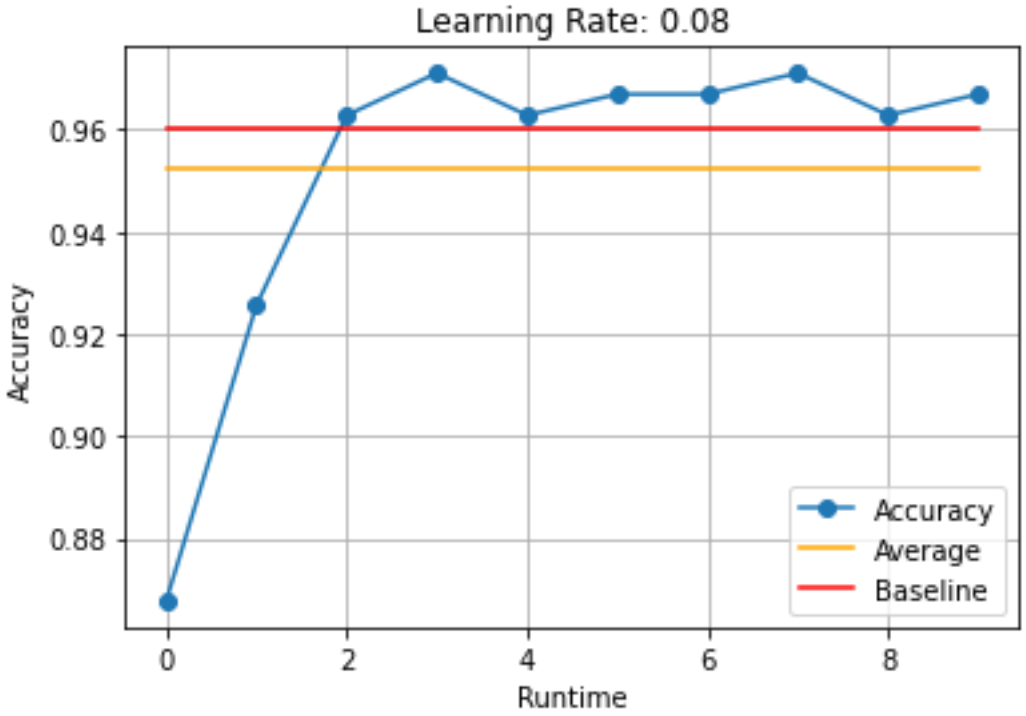
Use the number of hidden layers that achieves highest average accuracy in Q2. In this step, try 3 different learning rate values from the given interval. Run MLP 10 times for each case, plot runtime vs. accuracy with average line and updated baseline performance.

```
# 0.08 Learning Rate
model_lr_1 = keras.models.Sequential([
   Dense(5, activation='sigmoid', input_shape=(5,)),
   Dense(4, activation='sigmoid')
 ])

mod-
el_lr_1.compile(loss='sparse_categorical_crossentropy', optimizer=keras.optimizer
s.Adam(learning_rate=0.08), metrics=['accuracy'])

baseline = [0.96, 0.96, 0.96, 0.96, 0.96, 0.96, 0.96, 0.96, 0.96, 0.96]

h_total_train=np.empty((0,10))
h_total_val=np.empty((0,10))
total_acc=[]
number_of_runs=10
for i in range(number_of_runs):
  h = model_lr_1.fit(X_train, y_train, epochs=1000, batch_size=32, validation_spl
it=0.25, verbose=0)
  h_total_train=np.append(h_total_train,np.array([h.history['accuracy']]),axis=0)
  h_total_val=np.append(h_total_val,np.array([h.history['val_accuracy']]),axis=0)
  total_acc.append(model_lr_1.evaluate(X_test,y_test, verbose=0)[1])

for i in range(number_of_runs):
  f=plt.subplot(2,5,i+1)
  f.figure.set_size_inches(30,10)
  plt.plot(h_total_train[i])
  plt.plot(h_total_val[i],'r')
  plt.legend(['train acc', 'val acc'])
  plt.title("run # %d"%(i+1))
  plt.xlabel("# of epochs")
```

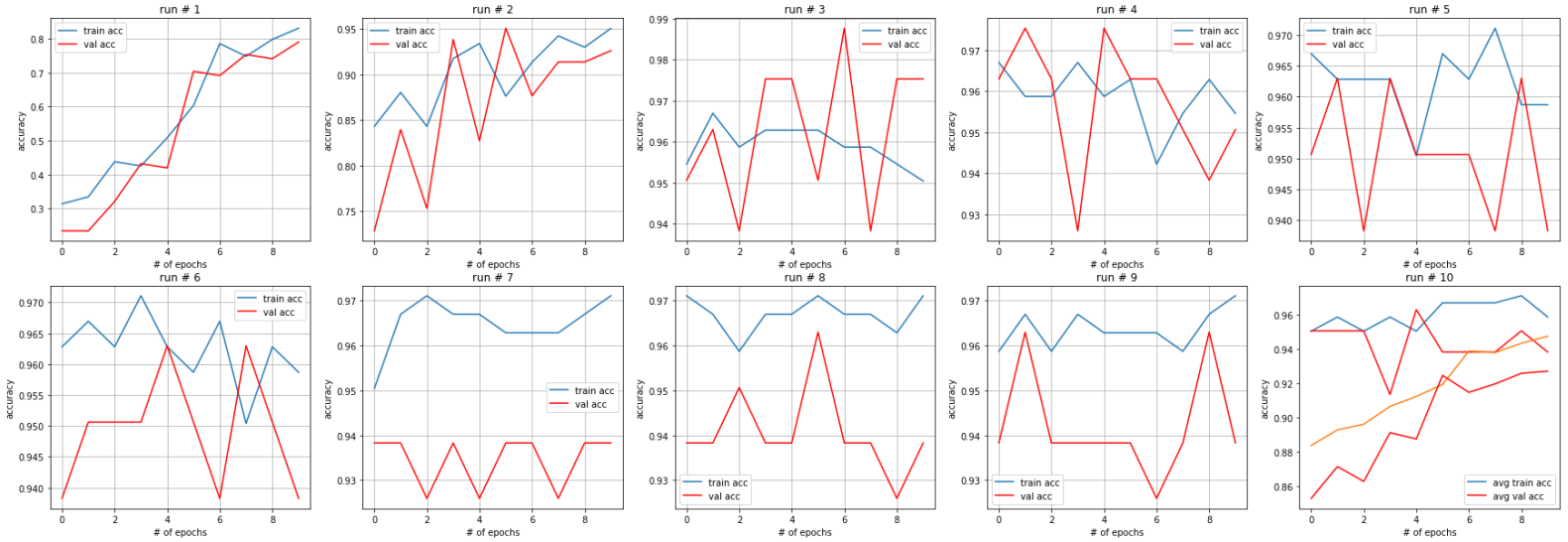
```
  plt.ylabel("accuracy")
  plt.grid()

h_avg_train=np.average(h_total_train,axis=0)
h_avg_val=np.average(h_total_val,axis=0)
plt.plot(h_avg_train)
plt.plot(h_avg_val, 'r')
plt.legend(['avg train acc', 'avg val acc'])
plt.xlabel("# of epochs")
plt.ylabel("accuracy")
plt.grid()
plt.show()

print('Test accuracy: %.2f %%'%(100*mean(total_acc)))

ten_runs_train=[]
ten_runs_val=[]
for i in range(number_of_runs):
  ten_runs_train.append(h_total_train[i][-1])
  ten_runs_val.append(h_total_val[i][-1])

plt.plot(ten_runs_train, marker='o')
plt.plot(range(0,10),[h_avg_train[-1]]*10,color='orange')
plt.plot(range(0,10),baseline,'r')
plt.legend(['Accuracy', 'Average', 'Baseline'])
plt.xlabel("Runtime")
plt.ylabel("Accuracy")
plt.title("Learning Rate: 0.08")
plt.grid()
plt.show()
```

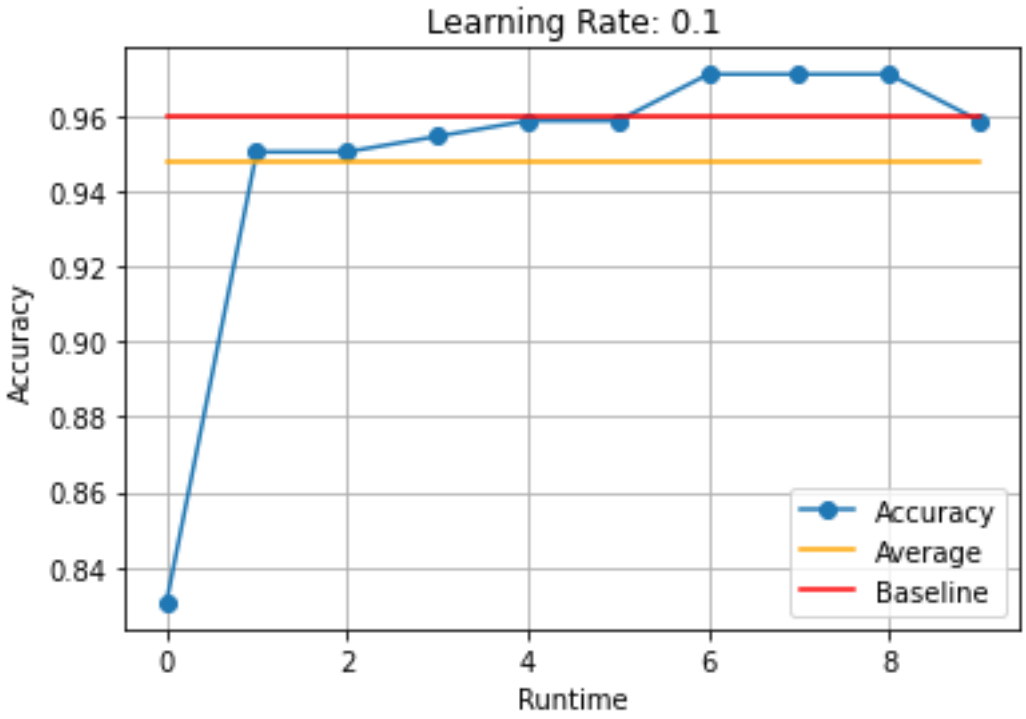
Test accuracy: 95.88 %



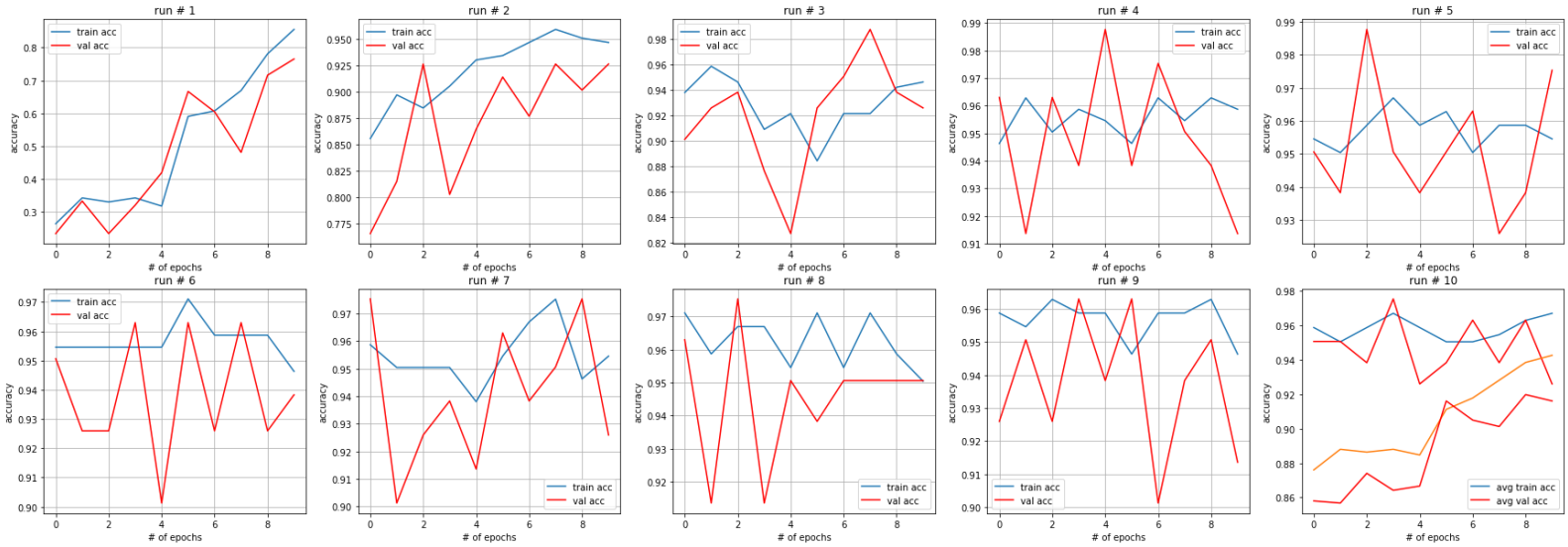


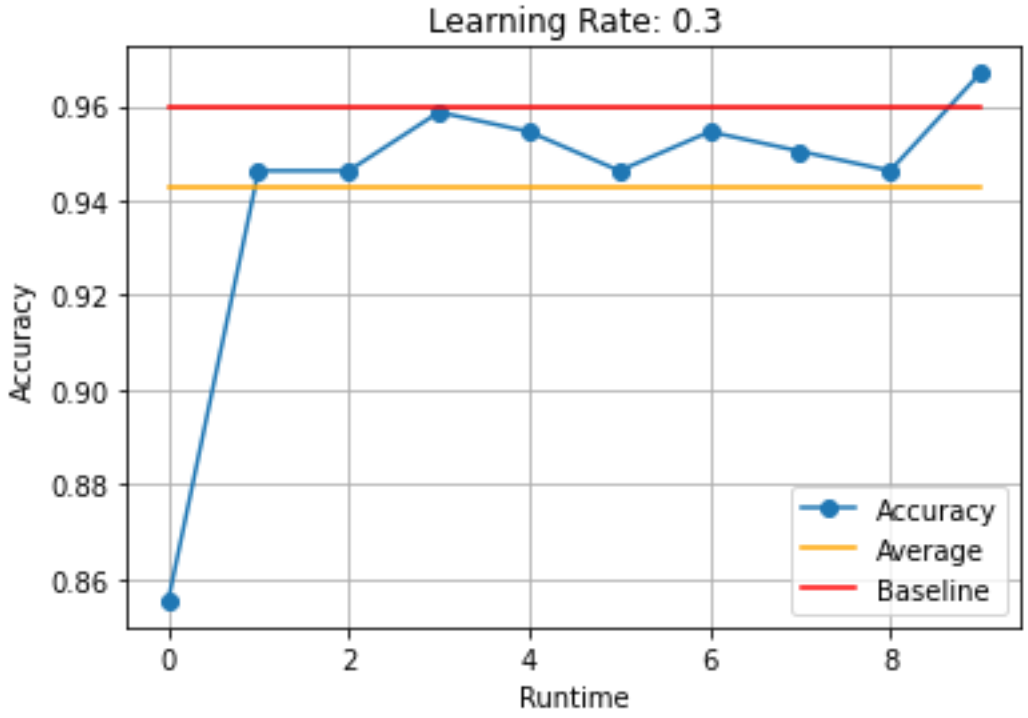
Test accuracy: 94.63 %





Test accuracy: 96.13 %





2. Train your model with final parameters. Display the training curve and confusion matrix with accuracy value.

```
# 0.08 Learning Rate
# 1 Hidden Layer
# Sigmoid Activation Function

best_model = keras.models.Sequential([
   Dense(5, activation='sigmoid', input_shape=(5,)),
   Dense(4, activation='sigmoid')
 ])

best_model.compile(loss='sparse_categorical_crossentropy', optimizer=keras.optimi
zers.Adam(learning_rate=0.08), metrics=['accuracy'])

h_total_train=np.empty((0,10))
h_total_val=np.empty((0,10))
total_acc=[]
number_of_runs=10
for i in range(number_of_runs):
  h = best_model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split
=0.25, verbose=0)
  h_total_train=np.append(h_total_train,np.array([h.history['accuracy']]),axis=0)
  h_total_val=np.append(h_total_val,np.array([h.history['val_accuracy']]),axis=0)
  total_acc.append(best_model.evaluate(X_test,y_test, verbose=0)[1])

for i in range(number_of_runs):
  f=plt.subplot(2,5,i+1)
  f.figure.set_size_inches(30,10)
  plt.plot(h_total_train[i])
  plt.plot(h_total_val[i],'r')
  plt.legend(['train acc', 'val acc'])
  plt.title("run # %d"%(i+1))
  plt.xlabel("# of epochs")
  plt.ylabel("accuracy")
  plt.grid()
```

```
h_avg_train=np.average(h_total_train,axis=0)
h_avg_val=np.average(h_total_val,axis=0)
plt.plot(h_avg_train)
plt.plot(h_avg_val, 'r')
plt.legend(['avg train acc', 'avg val acc'])
plt.xlabel("# of epochs")
plt.ylabel("accuracy")
plt.grid()
plt.show()

ten_runs_train=[]
ten_runs_val=[]
for i in range(number_of_runs):
  ten_runs_train.append(h_total_train[i][-1])
  ten_runs_val.append(h_total_val[i][-1])

plt.plot(ten_runs_train, marker='o')
plt.plot(range(0,10),[h_avg_train[-1]]*10,color='orange')

plt.legend(['Accuracy', 'Average'])
plt.xlabel("Runtime")
plt.ylabel("Accuracy")
plt.title("Activation Function: Sigmoid, Hidden Layers: 1, Learning Rate: 0.08")
plt.grid()
plt.show()

print('Test accuracy: %.2f %%'%(100*mean(total_acc)))

y_pred = np.argmax(best_model.predict(X_test),axis=1)

from sklearn.metrics import confusion_matrix

confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)
```

```
Test accuracy: 95.00 %
```

```
[[10  1  0  0]
 [ 0 25  1  0]
 [ 0  0 22  0]
 [ 0  0  0 21]]
```