In [104…

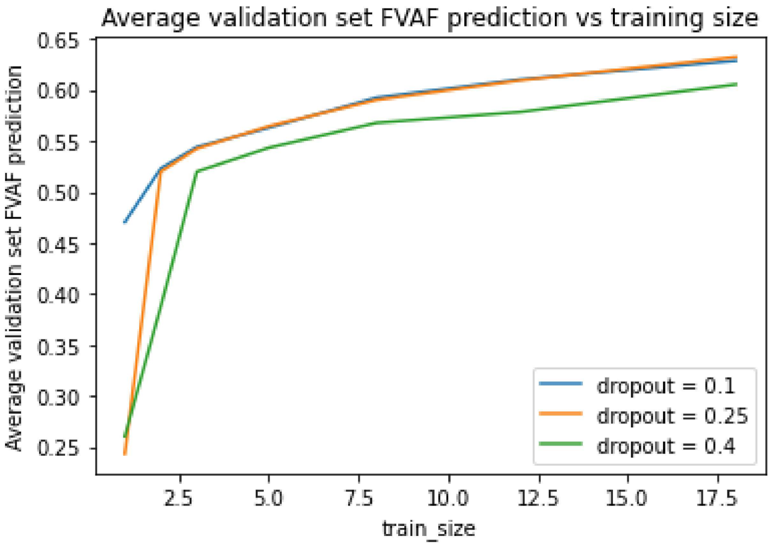
```
import matplotlib.pyplot as plt
import tensorflow as tf
import pandas as pd
import numpy as np
from tensorflow import keras
import os
import fnmatch
import time
import pickle
from tensorflow.keras.layers import InputLayer, Dense
from tensorflow.keras.models import Sequential
from scipy import stats

#opening the file
file_1 = open("bmi_dataset.pkl", "rb")
f_open_1 = pickle.load(file_1)
file_1.close()

#print(list(f_open_1.keys()))

# finding the actual testing labels
ins = f_open_1['MI']
Nfolds = len(ins)
folds_testing = (np.array([Nfolds-1]) + 0) % Nfolds
outs = f_open_1['torque']
outs_testing = np.concatenate(np.take(outs, folds_testing))
actual_testing = outs_testing[:,[0]]

# Getting the predicted testing labels
predict_testing = f_open_2['predict_testing']

## Getting the timestamp for predicted labels
timestamp = f_open_2['time_testing']


new_results = []

def read_all_rotations(dirname, filebase):
    '''Read results from dirname from files matching filebase'''

    # The set of files in the directory
    files = fnmatch.filter(os.listdir(dirname), filebase)
    files.sort()
    results = []


    # Loop over matching files
    for f in files:
        fp = open("%s/%s"%(dirname,f), "rb")
        r = pickle.load(fp)
        fp.close()
        results.append(r)
    return results

# matching the files
train = [1, 2, 3, 5, 8, 12, 18]
dropout = [0.1, 0.25, 0.4]
drop_new_res = []
```

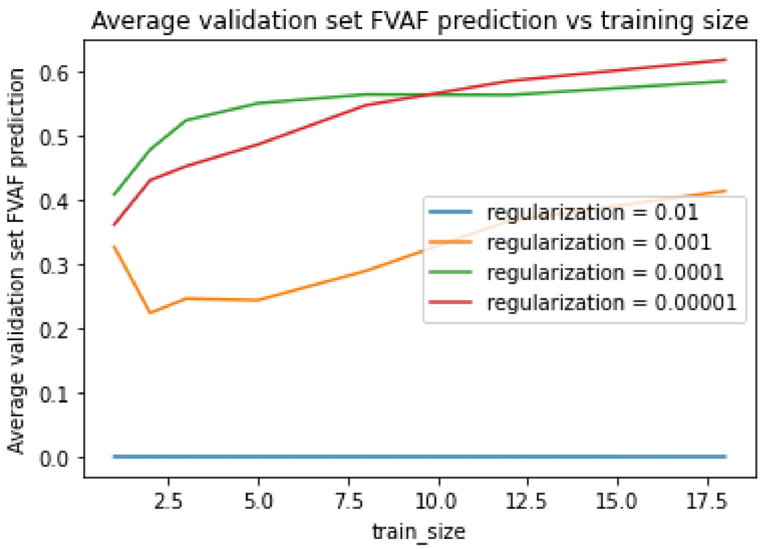
```
for d in dropout:
    for t in train:
        filebase = "bmi_torque_1_hidden_200_100_50_25_10_5_JI_Ntraining_"+str(t)+"_rot
        new_results.append( read_all_rotations("results_1", filebase))
    drop_new_res.append(new_results)
    new_results = []

avg_train = []
avg_validate = []

#print(len(drop_new_res[0][0][0]))

#temp_1 = []
temp_2 = []

res_drop = []
# calculating the average
for k in range(len(dropout)):
    avg_validate = []
    for i in range(len(train)):
        for j in range(len(drop_new_res[0][0][0])):
            temp_2.append(np.mean(drop_new_res[k][i][j]['predict_validation_eval'][1])

        avg_validate.append(sum(temp_2)/len(temp_2))
        temp_2 = []
    res_drop.append(avg_validate)
    #print(len(avg_validate))
#print(res_drop)

#plotting figure 2
for k in range(len(dropout)):
    #plt.plot(train,avg_train)
    plt.plot(train,res_drop[k])


plt.ylabel('Average validation set FVAF prediction')
plt.xlabel('train_size')
plt.title('Average validation set FVAF prediction vs training size')
plt.legend(['dropout = 0.1','dropout = 0.25', 'dropout = 0.4'])
#saving the figure2
plt.savefig("figure2.png")
plt.show()
plt.close()



#regularization
new_results = []
regularization = [0.01, 0.001, 0.0001, 0.00001]
regu_new_res = []
for r in regularization:
    for t in train:
        filebase = "bmi_torque_1_hidden_200_100_50_25_10_5_JI_Ntraining_"+str(t)+"_rot
        new_results.append( read_all_rotations("results_2", filebase))
    regu_new_res.append(new_results)
    new_results = []

avg_train = []
avg_validate = []
```

```
#print(len(drop_new_res[0][0][0]))

#temp_1 = []
temp_2 = []

res_regu = []
# calculating the average
for k in range(len(regularization)):
    avg_validate = []
    for i in range(len(train)):
        for j in range(len(regu_new_res[0][0][0])):
            temp_2.append(np.mean(regu_new_res[k][i][j]['predict_validation_eval'][1])

        avg_validate.append(sum(temp_2)/len(temp_2))
        temp_2 = []
    res_regu.append(avg_validate)
    #print(len(avg_validate))
#print(res_drop)

#plotting figure 3
for k in range(len(regularization)):
    #plt.plot(train,avg_train)
    plt.plot(train,res_regu[k])


plt.ylabel('Average validation set FVAF prediction')
plt.xlabel('train_size')
plt.title('Average validation set FVAF prediction vs training size')
plt.legend(['regularization = 0.01','regularization = 0.001', 'regularization = 0.0001
#saving the figure3
plt.savefig("figure3.png")
plt.show()
plt.close()

#converting result list into array
drop_array = np.array(res_drop)
regu_array = np.array(res_regu)

#finding the maximum
drop_max = np.argmax(drop_array, axis=0)
regu_max = np.argmax(regu_array, axis=0)

drop_dic = {}
regu_dic = {}

for i in range(len(drop_max)):
    drop_dic[train[i]] = dropout[drop_max[i]]

#best parameters
print("The best dropout parameter for the different training sizes are:\n")
print(drop_dic)

for i in range(len(regu_max)):
    regu_dic[train[i]] = regularization[regu_max[i]]

print("The best regularization parameter for the different training sizes are:\n")
print(regu_dic)

#mean test performance for best hyperparameters
temp_2 = []
```

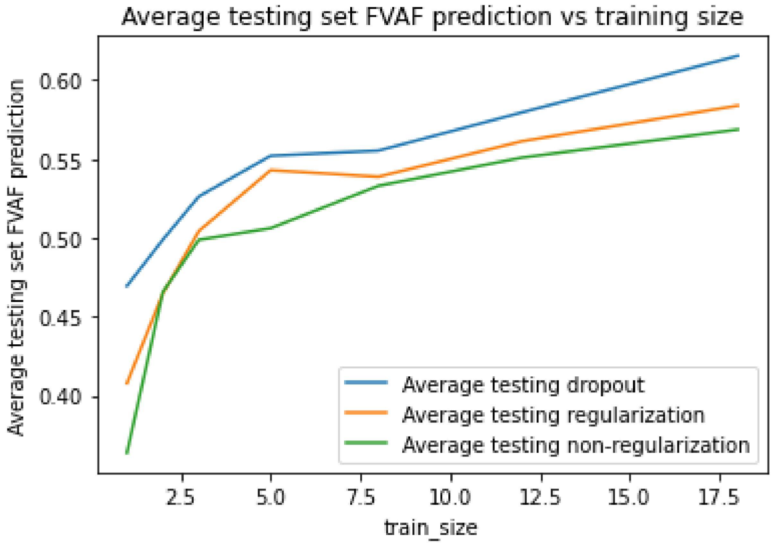
```
avg_test_drop = []
for i in range(len(drop_max)):
    for j in range(len(drop_new_res[0][0][0])):
        temp_2.append(np.mean(drop_new_res[drop_max[i]][i][j]['predict_testing_eval'][
    avg_test_drop.append(sum(temp_2)/len(temp_2))
    temp_2 = []


temp_2 = []
avg_test_regu = []
for i in range(len(regu_max)):
    for j in range(len(regu_new_res[0][0][0])):
        temp_2.append(np.mean(regu_new_res[regu_max[i]][i][j]['predict_testing_eval'][
    avg_test_regu.append(sum(temp_2)/len(temp_2))
    temp_2 = []

#mean test performance for non-regularized result
new_res = []
for t in train:
    filebase = "bmi_torque_1_hidden_200_100_50_25_10_5_JI_Ntraining_"+str(t)+"_rotatio
    new_res.append( read_all_rotations("results", filebase))



avg_test_non = []

temp_2 = []

for i in range(len(train)):
    for j in range(len(new_res[0][0])):
        temp_2.append(np.mean(new_res[i][j]['predict_testing_eval'][1]))

    avg_test_non.append(sum(temp_2)/len(temp_2))
    temp_2 = []

#figure 4

plt.plot(train, avg_test_drop)
plt.plot(train, avg_test_regu)
plt.plot(train, avg_test_non)


plt.ylabel('Average testing set FVAF prediction')
plt.xlabel('train_size')
plt.title('Average testing set FVAF prediction vs training size')
plt.legend(['Average testing dropout','Average testing regularization', 'Average testi
#saving the figure4
plt.savefig("figure4.png")
plt.show()
plt.close()

#p- values for training size 1
test_drop_1 = []
for j in range(len(drop_new_res[0][0][0])):
    test_drop_1.append(drop_new_res[drop_max[0]][0][j]['predict_testing_eval'][1])

test_reg_1 = []
for j in range(len(regu_new_res[0][0][0])):
    test_reg_1.append(regu_new_res[regu_max[0]][0][j]['predict_testing_eval'][1])
```

```
test_non_1 = []
for j in range(len(new_res[0][0])):
        test_non_1.append(new_res[0][j]['predict_testing_eval'][1])

test_drop_1_array = np.array(test_drop_1)
test_reg_1_array = np.array(test_reg_1)
test_non_1_array = np.array(test_non_1)

d, dr_1 = stats.ttest_rel(test_drop_1_array, test_reg_1_array)
r, rn_1 = stats.ttest_rel(test_reg_1_array, test_non_1_array)
n, nd_1 = stats.ttest_rel(test_drop_1_array,test_non_1_array)

print("the p-value for dropout and regularization for training size 1: %f\n"%dr_1)
print("the p-value for regularization and non-regularization for training size 1: %f\
print("the p-value for dropout and non-regularization for training size 1: %f\n"%nd_1)

#p- values for training size 18
test_drop_18 = []
for j in range(len(drop_new_res[0][0][0])):
    test_drop_18.append(drop_new_res[drop_max[-1]][-1][j]['predict_testing_eval'][1])

test_reg_18 = []
for j in range(len(regu_new_res[0][0][0])):
    test_reg_18.append(regu_new_res[regu_max[-1]][-1][j]['predict_testing_eval'][1])


test_non_18 = []
for j in range(len(new_res[0][0])):
        test_non_18.append(new_res[-1][j]['predict_testing_eval'][1])

test_drop_18_array = np.array(test_drop_18)
test_reg_18_array = np.array(test_reg_18)
test_non_18_array = np.array(test_non_18)

d, dr_18 = stats.ttest_rel(test_drop_18_array, test_reg_18_array)
r, rn_18 = stats.ttest_rel(test_reg_18_array, test_non_18_array)
n, nd_18 = stats.ttest_rel(test_drop_18_array,test_non_18_array)

print("the p-value for dropout and regularization for training size 18: %f\n"%dr_18)
print("the p-value for regularization and non-regularization for training size 18: %f\
print("the p-value for dropout and non-regularization for training size 18: %f\n"%nd_1
```





```
The best dropout parameter for the different training sizes are:

{1: 0.1, 2: 0.1, 3: 0.1, 5: 0.25, 8: 0.1, 12: 0.1, 18: 0.25}
The best regularization parameter for the different training sizes are:

{1: 0.0001, 2: 0.0001, 3: 0.0001, 5: 0.0001, 8: 0.0001, 12: 1e-05, 18: 1e-05}
```



```
the p-value for dropout and regularization for training size 1: 0.001221

the p-value for regularization and non-regularization for training size 1: 0.057156

the p-value for dropout and non-regularization for training size 1: 0.002473

the p-value for dropout and regularization for training size 18: 0.020630

the p-value for regularization and non-regularization for training size 18: 0.278191

the p-value for dropout and non-regularization for training size 18: 0.000329
```

In [ ]:

In [ ]: