```
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
import matplotlib.patches as mpatches

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
```

```
#shallow network

filebase =
"image_Csize_3_3_1_Cfilters_16_32_64_Pool_2_2_1_hidden_50_20_drop_0.500_L2_0.000100_LR_0.0
01000_ntrain_03_rot_*_results.pkl"

new_res = read_all_rotations("results", filebase)


#deep network

#filebase1 =
"image_Csize_3_3_Cfilters_16_16_Pool_2_2_hidden_35_30_25_20_drop_0.500_L2_0.000100_LR_0.00
1000_ntrain_03_rot_*_results.pkl"

filebase1 =
"image_Csize_3_1_Cfilters_16_32_Pool_2_1_hidden_50_40_30_20_drop_0.500_L2_0.000100_LR_0.00
1000_ntrain_03_rot_*_results.pkl"

new_res1 = read_all_rotations("results", filebase1)


print(new_res[0]['predict_testing_eval'])

#figure1

for i in range(len(new_res)):

   plt.plot(new_res[i]['history']['val_loss'],linestyle='dashed',label = 'shallow_rotation_'+str(i+1))

for i in range(len(new_res1)):

   plt.plot(new_res1[i]['history']['val_loss'],label = 'deep_rotation_'+str(i+1))


plt.ylabel('validation loss')

plt.xlabel('epochs')

plt.title('validation loss vs Epochs for each rotation for different networks')

plt.legend()

plt.savefig("validation_loss.png")

plt.show()

plt.close()
```

```
print(len(new_res))


#figure2
for i in range(len(new_res)):
    plt.plot(new_res[i]['history']['val_categorical_accuracy'],linestyle='dashed',label =
'shallow_rotation_'+str(i+1))
for i in range(len(new_res)):
    plt.plot(new_res1[i]['history']['val_categorical_accuracy'],label = 'deep_rotation_'+str(i+1))


plt.ylabel('validation accuracy')
plt.xlabel('epochs')
plt.title('validation accuracy vs Epochs for each rotation for different networks')
plt.legend()
plt.savefig("validation_accuracy.png")
plt.show()
plt.close()


#figure3
for i in range(len(new_res)):
    plt.hist(new_res[i]['predict_testing_eval'][1], alpha=0.5, color='green',label='shallow_network')
for i in range(len(new_res1)):
    plt.hist(new_res1[i]['predict_testing_eval'][1], alpha=0.5, color='red',label='deep_network')
plt.xlabel('testing performance')
plt.title('Histogram on testing performance for different networks')
a = mpatches.Patch(color='green', label='shallow_network')
b = mpatches.Patch(color='red', label='deep_network')
plt.legend(handles=[a,b])
plt.savefig("histogram_accuracy.png")
plt.show()
```

plt.close()

1. What is the outline of your network architecture?

shallow networks = 4,596,317 parameters, 3 convolution layers, 4 inception layers, 2 dense layers

deep networks = 22,561,139 parameters, 2 convolution layers, 2 inception layers, 4 dense layers

2. Which model (shallow or deep) turned out to work better? Did you have to adjust hyper-parameters between the two, other than network structure?

The deep network worked better than shallow network. The parameters that were tuned was the rotation

range set to 30 and horizontal filp set to true

3. What can you conclude from the validation accuracy learning curves for each of the shallow and deep networks? How confident are you that you have created models that you can trust?

The deep networks did comparatively well than the shallow networks. But there is lot that can be done well

for getting a better validation accuracy since now based on the graphs we can see it goes all over the place.

4. Did your shallow or deep network perform better with respect to the test set?

The deep network network performed well with respect to the test set.

5. Did data augmentation improve or inhibit model performance with respect to the validation and test sets?

Yes the data augmentation looks to make the model performance more better .