Untitled3

November 29, 2018

```
In [15]: # -*- coding: utf-8 -*-
         """
         Created on Mon Nov 26 09:23:45 2018

         @author: aza8223
         """

         """Project2"""


         ################################################################################

         """Importing important libraries"""

         import numpy as np
         import pandas as pd
         from keras import layers
         from keras import optimizers
         import math
         import matplotlib.pyplot as plt
         from keras.optimizers import RMSprop

In [16]: """Preparing data"""
         data =[]
         if data:
             del data

         total_data = pd.read_csv("C:/Users/aza8223/OneDrive - University of Tulsa/to_be_transfe
         data = total_data[:,1:]

         """Check if there is nan (missing) data and replace them with their next data:"""
         """Here i have used while loop for the case when oreceding samples all nan replacement
         keeps going until get reasonable neighbor value"""
         data = pd.DataFrame(data=data)
         while 1:
             for j, kays in enumerate(data.loc[0,:]):
                 for i, kay in enumerate(data.loc[:,0]):
                     if math.isnan(data.loc[i,j]):
```

```
                data.loc[i,j]=data.loc[i+1,j]
                print("sample ", i, "feature", j, " was missing and replaced by its nex
    if not data.isnull().any().any():
        break
data = np.asarray(data).astype('float32')

"""Change true and false to 1 and 0"""
for j, rain in enumerate(data[:,3]):
    if data[j, 3]==True:
        data[j,3]=1
    else:
        data[j,3]=0

data = data[:,:3] #If it rains or not is not important feature for the determination
#of amount of rain.
data = np.asarray(data).astype('float32')

"""Creating descriptive and target features"""
num_data = len(data)
output_size = 7 #Days to be predicted. They are fixed
input_size = 7 #Sequence of days to be descriptive feature. You can modify it
# as given in the problem: 1 day, 7 days, 14 days, 1 months.


"""Create data descriptime sequential features with the shape of sample*times*features"
data_feat = np.zeros((num_data-(output_size+input_size),input_size,len(data[0])))
data_label = np.zeros((num_data-(output_size+input_size),output_size))
for i in range(num_data - (output_size+input_size)):
    data_feat[i] = data[i:i+input_size]
    data_label[i] = data[i+input_size:i+input_size+output_size,0]

"""Seperating data into dry and wet days"""
"""
To do so, i calculated mean of each output (7days that to be predicted)
then i compared that output with mean of all labels, and thus i devided my data
for dry week and wet week
"""
mean_each_output = data_label[:,:].mean(axis=1)
mean_all_data = np.nanmean(mean_each_output)

positive_data = []
positive_label = []
negative_data = []
negative_label = []

for i in range(len(data_label)):
    if mean_each_output[i]<=mean_all_data:
        negative_data.append(data_feat[i])
        negative_label.append(data_label[i])
    else:
        positive_data.append(data_feat[i])
        positive_label.append(data_label[i])

positive_data = np.asarray(positive_data).astype('float32')
positive_data_part1 = positive_data[:round(len(positive_data)/3)]
positive_data_part2 = positive_data[round(len(positive_data)/3):round(2*len(positive_da
positive_data_part3 = positive_data[round(2*len(positive_data)/3):]

positive_label = np.asarray(positive_label).astype('float32')
positive_label_part1 = positive_label[:round(len(positive_data)/3)]
positive_label_part2 = positive_label[round(len(positive_data)/3):round(2*len(positive_
positive_label_part3 = positive_label[round(2*len(positive_data)/3):]

negative_data = np.asarray(negative_data).astype('float32')
negative_data_part1 = negative_data[:round(len(negative_data)/3)]
negative_data_part2 = negative_data[round(len(negative_data)/3):round(2*len(negative_da
negative_data_part3 = negative_data[round(2*len(negative_data)/3):]

negative_label = np.asarray(negative_label).astype('float32')
negative_label_part1 = negative_label[:round(len(negative_data)/3)]
negative_label_part2 = negative_label[round(len(negative_data)/3):round(2*len(negative_
negative_label_part3 = negative_label[round(2*len(negative_data)/3):]

"""Create training, test, validation data and labels using 1/3 partion of both
negative and positive sets:"""

import itertools
training_data = []
for item in itertools.chain(positive_data_part1,negative_data_part1):
    training_data.append(item)

training_labels = []
for item in itertools.chain(positive_label_part1,negative_label_part1):
    training_labels.append(item)

test_data = []
for item in itertools.chain(positive_data_part2,negative_data_part2):
    test_data.append(item)

test_labels = []
for item in itertools.chain(positive_label_part2,negative_label_part2):
    test_labels.append(item)

val_data = []
for item in itertools.chain(positive_data_part3,negative_data_part3):
    val_data.append(item)
```

```
val_labels = []
for item in itertools.chain(positive_label_part3,negative_label_part3):
    val_labels.append(item)


training_data = np.asarray(training_data).astype('float32')
training_labels = np.asarray(training_labels).astype('float32')

test_data = np.asarray(test_data).astype('float32')
test_labels = np.asarray(test_labels).astype('float32')

val_data = np.asarray(val_data).astype('float32')
val_labels = np.asarray(val_labels).astype('float32')

"""Shuffle data and labels:"""

from random import shuffle

ind_list = [i for i in range(len(training_data))]
shuffle(ind_list)
training_data  = training_data[ind_list, :, :]
training_labels = training_labels[ind_list, :]

ind_list = [i for i in range(len(val_data))]
shuffle(ind_list)
val_data  = val_data[ind_list, :, :]
val_labels = val_labels[ind_list, :]

ind_list = [i for i in range(len(test_data))]
shuffle(ind_list)
test_data  = test_data[ind_list, :, :]
test_labels = test_labels[ind_list, :]
```

```
sample  18415 feature 0  was missing and replaced by its next samnple
sample  18416 feature 0  was missing and replaced by its next samnple
sample  21067 feature 0  was missing and replaced by its next samnple
sample  18415 feature 3  was missing and replaced by its next samnple
sample  18416 feature 3  was missing and replaced by its next samnple
sample  21067 feature 3  was missing and replaced by its next samnple
sample  18415 feature 0  was missing and replaced by its next samnple
sample  18415 feature 3  was missing and replaced by its next samnple
```

```
In [17]: #Normalize your all data based on mean std of your training data and training labels:
         mean = training_data[:,:,:].mean(axis=0)
         training_data[:,:,:] -= mean
         std = np.std(training_data[:,:,:],axis=0)
```

```
training_data[:,:,:] /= std

val_data[:,:,:] -= mean
val_data[:,:,:] /= std

test_data[:,:,:] -= mean
test_data[:,:,:] /= std

mean = training_labels[:,:].mean(axis=0)
training_labels[:,:] -= mean
std = np.std(training_labels[:,:],axis=0)
training_labels[:,:] /= std

val_labels[:,:] -= mean
val_labels[:,:] /= std

test_labels[:,:] -= mean
test_labels[:,:] /= std
```

In [18]:

```
"""Base case for each day and mean of mae"""
"""Here I took average of previous days as my predictor for the each day of the
next week. Therefore I have calculated mae for each day of the next week. To
be able to compare this mae with my models, since I predict them all together, and
therefore I have 1 mae for  model, I took average of all those mae in this base
model for each day and took mean of them. I will use this mean of mae of the days of
the next week to compare it with my models. However, at the last model, where
I use multiple output DAG model, I used mae of each day in my base model to compare
it with the loss of each day in that last model:"""
preds = np.mean(val_data[:, :, 0], axis=1)
day = np.zeros((val_labels.shape[1], val_labels.shape[0]))
mae_base1 = np.zeros((val_labels.shape[1],))
for i,j in enumerate(np.transpose(val_labels)):
    day[i] = val_labels[:,i]
    mae_base1[i] = np.nanmean(np.abs(preds - day[i]))
    print('normalized MAE of base model for day ', i+1, " is ", mae_base1[i])
    print('unnormalized MAE of base model for day ', i+1, " is ", mae_base1[i]*std[0])
mae_base_mean = mae_base1.mean()
print('mean of normalized MAE of base model of week ', " is ", mae_base_mean)
print('mean of unnormalized MAE of base model of week ', " is ", mae_base_mean*std[0])



"""Base model2: This is just my own opinion, but I ll not compare my models with this m
In the following base model2, I choose my target  not as each dy of next week but avera
of them. So I found mae between average precipitation of previous days as predictor of
average precipitation. This result showed 10 percent of mae. Compared to the base
model given above it is higher but it doesnt show that this is good predictor of
each day of next week, but it is good model to predict average precipitation of the
```

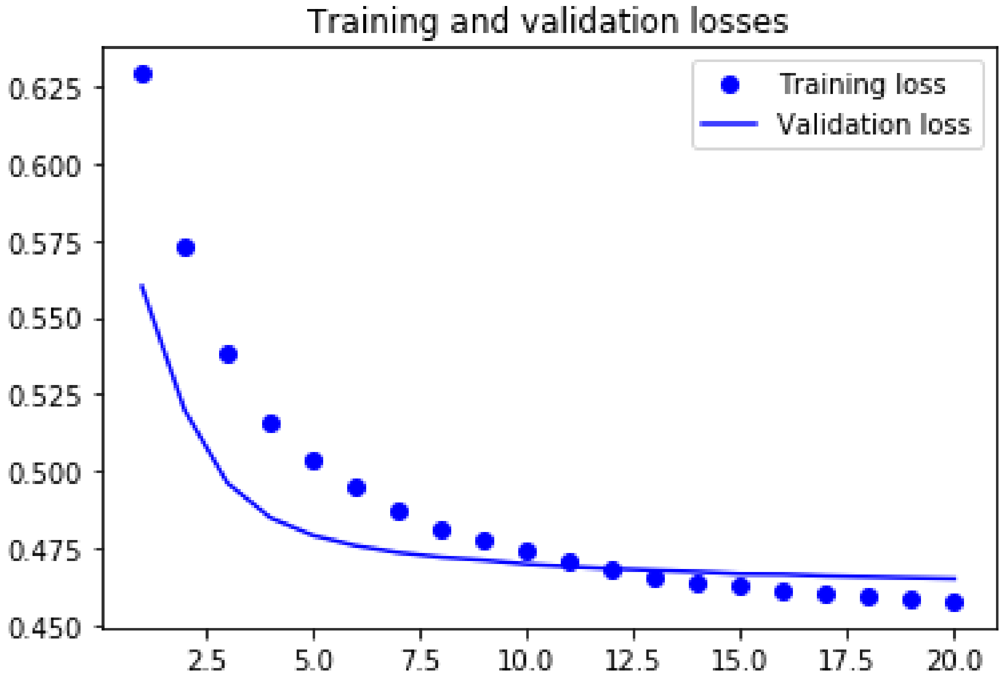
```
         next week:"""
         preds = np.mean(val_data[:, :, 0], axis=1)
         week_data = np.mean(val_labels[:,:],axis=1)
         mae_base2 = np.nanmean(np.abs(preds - week_data))
         print('normalized MAE of base2 model is ', mae_base2)
         print('unnormalized MAE of base2 model is ', mae_base2*std[0])
```

```
normalized MAE of base model for day  1  is  0.5893916490332757
unnormalized MAE of base model for day  1  is  0.13748106907202862
normalized MAE of base model for day  2  is  0.6158605915959213
unnormalized MAE of base model for day  2  is  0.14365519543891442
normalized MAE of base model for day  3  is  0.6282546054119308
unnormalized MAE of base model for day  3  is  0.146546214122863
normalized MAE of base model for day  4  is  0.6354590758521732
unnormalized MAE of base model for day  4  is  0.14822672367851578
normalized MAE of base model for day  5  is  0.6384783752504914
unnormalized MAE of base model for day  5  is  0.14893100326885297
normalized MAE of base model for day  6  is  0.6422891159124167
unnormalized MAE of base model for day  6  is  0.14981989387498396
normalized MAE of base model for day  7  is  0.646235350870764
unnormalized MAE of base model for day  7  is  0.15074038978254028
mean of normalized MAE of base model of week   is  0.6279955377038533
mean of unnormalized MAE of base model of week   is  0.146485784176957
normalized MAE of base2 model is  0.4666682
unnormalized MAE of base2 model is  0.10885468
```

```
In [19]: """1: Training and evaluating a densely connected model"""
         """I have tried different kind of architectures hidden units etc, but found this
         useful since it does not overfit and I got lower loss - 0.1015 (unnormilized)"""
         from keras.models import Sequential
         model = Sequential()
         model.add(layers.Flatten(input_shape=(input_size, training_data.shape[-1])))
         model.add(layers.Dense(64,activation='tanh'))
         model.add(layers.Dropout(0.3))
         model.add(layers.Dense(32,activation='tanh'))
         model.add(layers.Dense(output_size,activation='tanh'))


         """COMPILE YOUR MODEL"""
         model.compile(optimizer=optimizers.RMSprop(lr=1e-4), loss='mae')


         """TRAINING YOUR MODEL"""
         epoch_size = 20
         batch_size = 32
         history = model.fit(training_data,
                             training_labels,
                             epochs=epoch_size,
                             batch_size=batch_size,
                             validation_data = (val_data, val_labels))


         """Plotting results"""
         loss = history.history['loss']
         val_loss = history.history['val_loss']
         epochs = range(1, len(loss) + 1)
         plt.figure()
         plt.plot(epochs, loss, 'bo', label='Training loss')
         plt.plot(epochs, val_loss, 'b', label='Validation loss')
         plt.title('Training and validation losses')
         plt.legend()
         plt.show()


         """PREDICTION - TESTING DATA"""
         test_loss = model.evaluate(test_data, test_labels)
         print('normalized test_loss:', test_loss)
         print('unnormalized test_loss:', test_loss*std[0])

         """Save your model:"""
         model.save('C:/Users/aza8223/OneDrive - University of Tulsa/to_be_transfered/python/Pro
```

```
Train on 8512 samples, validate on 8512 samples
Epoch 1/20
8512/8512 [==============================] - 1s 127us/step - loss: 0.6292 - val_loss: 0.5600
Epoch 2/20
8512/8512 [==============================] - 0s 36us/step - loss: 0.5735 - val_loss: 0.5198
Epoch 3/20
8512/8512 [==============================] - 0s 36us/step - loss: 0.5385 - val_loss: 0.4961
Epoch 4/20
8512/8512 [==============================] - 0s 34us/step - loss: 0.5162 - val_loss: 0.4848
Epoch 5/20
8512/8512 [==============================] - 0s 35us/step - loss: 0.5033 - val_loss: 0.4790
Epoch 6/20
8512/8512 [==============================] - 0s 37us/step - loss: 0.4947 - val_loss: 0.4756
Epoch 7/20
8512/8512 [==============================] - 0s 36us/step - loss: 0.4868 - val_loss: 0.4734
Epoch 8/20
8512/8512 [==============================] - 0s 34us/step - loss: 0.4814 - val_loss: 0.4719
Epoch 9/20
8512/8512 [==============================] - 0s 34us/step - loss: 0.4776 - val_loss: 0.4709
Epoch 10/20
8512/8512 [==============================] - 0s 37us/step - loss: 0.4744 - val_loss: 0.4696
Epoch 11/20
8512/8512 [==============================] - 0s 35us/step - loss: 0.4707 - val_loss: 0.4688
```

```
Epoch 12/20
8512/8512 [==============================] - 0s 34us/step - loss: 0.4679 - val_loss: 0.4682
Epoch 13/20
8512/8512 [==============================] - 0s 36us/step - loss: 0.4657 - val_loss: 0.4677
Epoch 14/20
8512/8512 [==============================] - 0s 35us/step - loss: 0.4640 - val_loss: 0.4672
Epoch 15/20
8512/8512 [==============================] - 0s 35us/step - loss: 0.4625 - val_loss: 0.4665
Epoch 16/20
8512/8512 [==============================] - 0s 36us/step - loss: 0.4612 - val_loss: 0.4664
Epoch 17/20
8512/8512 [==============================] - 0s 34us/step - loss: 0.4605 - val_loss: 0.4659
Epoch 18/20
8512/8512 [==============================] - 0s 36us/step - loss: 0.4592 - val_loss: 0.4656
Epoch 19/20
8512/8512 [==============================] - 0s 36us/step - loss: 0.4584 - val_loss: 0.4652
Epoch 20/20
8512/8512 [==============================] - 0s 34us/step - loss: 0.4577 - val_loss: 0.4650
```

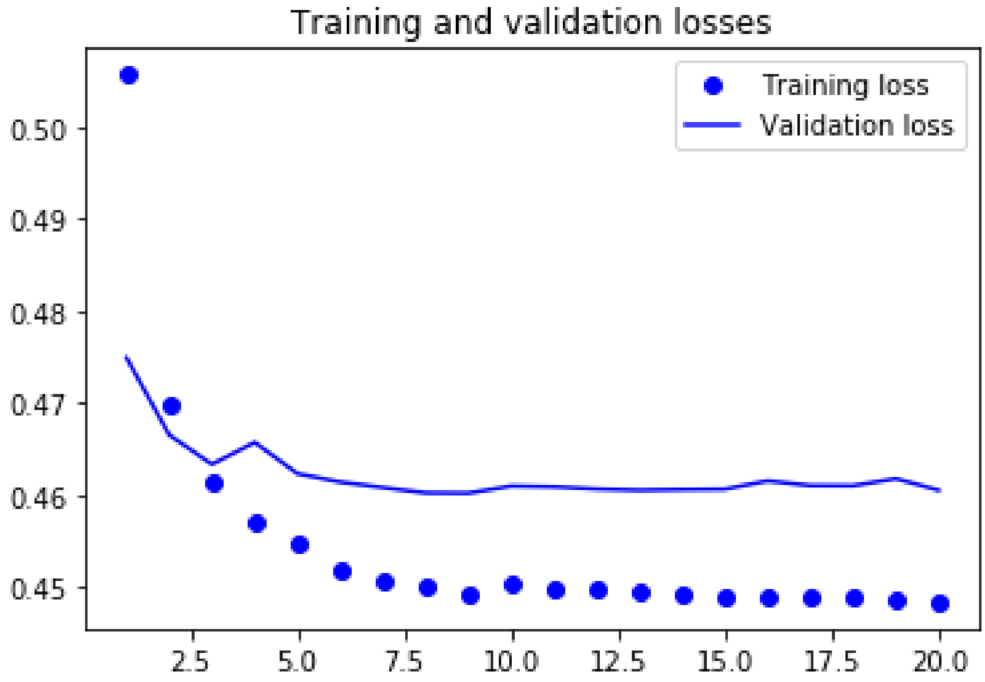


```
8513/8513 [==============================] - 0s 12us/step
normalized test_loss: 0.4356100456306682
unnormalized test_loss: 0.10161008366855616
```

```
In [20]: """2a: RNN"""
"""I have tried different dense model architecture but best one was this
which is 2nd dense with 32 hidden units"""
"""Dropout also helped to improve model. I kept playing with dropouts and
additional dropout layer until i get least loss"""
"""But when i rerun model it gives me different kind of test_loss values
even thoough i train the same model( between 18 and 48). that means our data is very un
therefore stochastig gradient method catch different local minimum each time"""
model = Sequential()
model.add(layers.GRU(32,
                     dropout=0.2,
                     recurrent_dropout=0.2,
                     input_shape=(None, training_data.shape[-1])))
model.add(layers.Dense(32,activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(output_size,activation='tanh'))

"""COMPILE YOUR MODEL"""
model.compile(optimizer=RMSprop(), loss='mae')


"""TRAINING YOUR MODEL"""
epoch_size = 20
batch_size = 32
history = model.fit(training_data,
                    training_labels,
                    epochs=epoch_size,
                    batch_size=batch_size,
                    validation_data = (val_data, val_labels))


"""Plotting results"""
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation losses')
plt.legend()
plt.show()


"""PREDICTION - TESTING DATA"""
test_loss = model.evaluate(test_data, test_labels)
print('normalized test_loss:', test_loss)
print('unnormalized test_loss:', test_loss*std[0])
```

```
"""Save your model:"""
model.save('C:/Users/aza8223/OneDrive - University of Tulsa/to_be_transfered/python/Pro
```

```
Train on 8512 samples, validate on 8512 samples
Epoch 1/20
8512/8512 [==============================] - 3s 298us/step - loss: 0.5057 - val_loss: 0.4749
Epoch 2/20
8512/8512 [==============================] - 1s 137us/step - loss: 0.4699 - val_loss: 0.4665
Epoch 3/20
8512/8512 [==============================] - 1s 134us/step - loss: 0.4613 - val_loss: 0.4633
Epoch 4/20
8512/8512 [==============================] - 1s 136us/step - loss: 0.4569 - val_loss: 0.4657
Epoch 5/20
8512/8512 [==============================] - 1s 133us/step - loss: 0.4546 - val_loss: 0.4623
Epoch 6/20
8512/8512 [==============================] - 1s 133us/step - loss: 0.4518 - val_loss: 0.4614
Epoch 7/20
8512/8512 [==============================] - 1s 138us/step - loss: 0.4507 - val_loss: 0.4608
Epoch 8/20
8512/8512 [==============================] - 1s 138us/step - loss: 0.4501 - val_loss: 0.4602
Epoch 9/20
8512/8512 [==============================] - 1s 134us/step - loss: 0.4493 - val_loss: 0.4602
Epoch 10/20
8512/8512 [==============================] - 1s 135us/step - loss: 0.4503 - val_loss: 0.4609
Epoch 11/20
8512/8512 [==============================] - 1s 134us/step - loss: 0.4497 - val_loss: 0.4609
Epoch 12/20
8512/8512 [==============================] - 1s 134us/step - loss: 0.4498 - val_loss: 0.4606
Epoch 13/20
8512/8512 [==============================] - 1s 135us/step - loss: 0.4494 - val_loss: 0.4605
Epoch 14/20
8512/8512 [==============================] - 1s 135us/step - loss: 0.4490 - val_loss: 0.4606
Epoch 15/20
8512/8512 [==============================] - 1s 133us/step - loss: 0.4490 - val_loss: 0.4606
Epoch 16/20
8512/8512 [==============================] - 1s 138us/step - loss: 0.4490 - val_loss: 0.4615
Epoch 17/20
8512/8512 [==============================] - 1s 136us/step - loss: 0.4488 - val_loss: 0.4610
Epoch 18/20
8512/8512 [==============================] - 1s 136us/step - loss: 0.4489 - val_loss: 0.4610
Epoch 19/20
8512/8512 [==============================] - 1s 135us/step - loss: 0.4487 - val_loss: 0.4617
Epoch 20/20
8512/8512 [==============================] - 1s 133us/step - loss: 0.4483 - val_loss: 0.4605
```



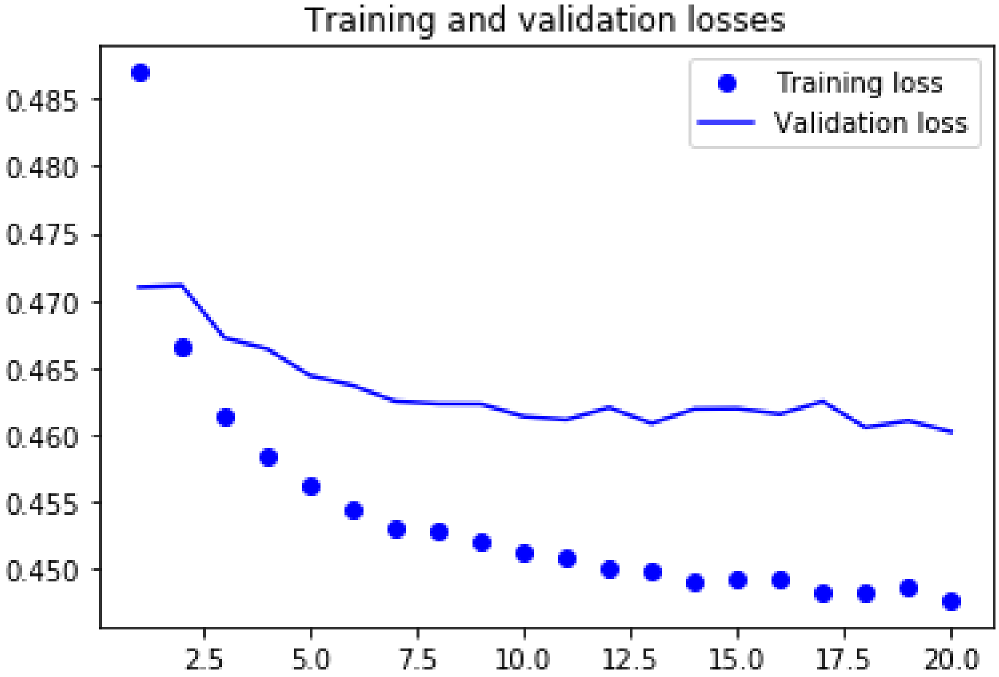
```
8513/8513 [==============================] - 0s 26us/step
normalized test_loss: 0.4303598094446692
unnormalized test_loss: 0.10038541737931418
```

```
In [21]: """2b: Training and evaluating a dropout-regularized, stacked GRU model"""


         from keras.models import Sequential
         from keras import layers
         from keras.optimizers import RMSprop
         model = Sequential()
         model.add(layers.GRU(32, activation='relu',
                              dropout=0.2,
                              recurrent_dropout=0.2,
                              return_sequences=True,
                              input_shape=(None, training_data.shape[-1])))
         model.add(layers.GRU(64, activation='relu',
                              dropout=0.2,
                              recurrent_dropout=0.25))
         model.add(layers.Dense(32, activation='relu'))
         model.add(layers.Dropout(0.3))
         model.add(layers.Dense(output_size, activation='tanh'))

         """COMPILE YOUR MODEL"""
```

```python
model.compile(optimizer=RMSprop(), loss='mae')


"""TRAINING YOUR MODEL"""
epoch_size = 20
batch_size = 32
history = model.fit(training_data,
                    training_labels,
                    epochs=epoch_size,
                    batch_size=batch_size,
                    validation_data = (val_data, val_labels))


"""Plotting results"""
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation losses')
plt.legend()
plt.show()


"""PREDICTION - TESTING DATA"""
test_loss = model.evaluate(test_data, test_labels)
print('normalized test_loss:', test_loss)
print('unnormalized test_loss:', test_loss*std[0])

"""Save your model:"""
model.save('C:/Users/aza8223/OneDrive - University of Tulsa/to_be_transfered/python/Pro
```

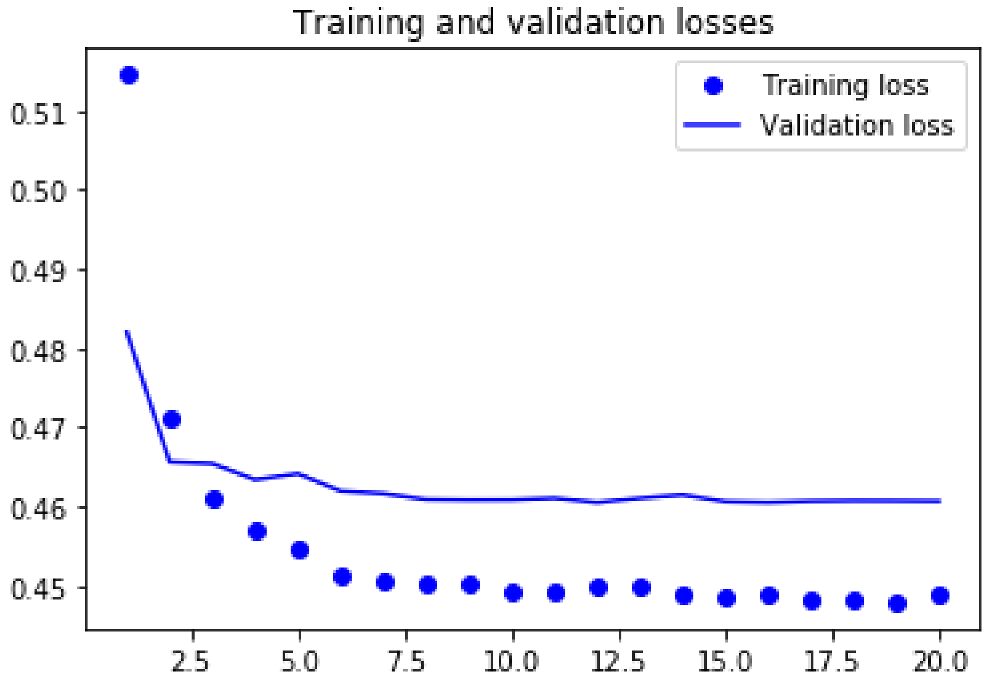
```
Train on 8512 samples, validate on 8512 samples
Epoch 1/20
8512/8512 [==============================] - 4s 492us/step - loss: 0.4870 - val_loss: 0.4710
Epoch 2/20
8512/8512 [==============================] - 2s 238us/step - loss: 0.4665 - val_loss: 0.4711
Epoch 3/20
8512/8512 [==============================] - 2s 235us/step - loss: 0.4614 - val_loss: 0.4672
Epoch 4/20
8512/8512 [==============================] - 2s 240us/step - loss: 0.4584 - val_loss: 0.4664
Epoch 5/20
8512/8512 [==============================] - 2s 244us/step - loss: 0.4561 - val_loss: 0.4644
Epoch 6/20
8512/8512 [==============================] - 2s 236us/step - loss: 0.4544 - val_loss: 0.4637
Epoch 7/20
8512/8512 [==============================] - 2s 238us/step - loss: 0.4529 - val_loss: 0.4625
```

```
Epoch 8/20
8512/8512 [==============================] - 2s 234us/step - loss: 0.4529 - val_loss: 0.4623
Epoch 9/20
8512/8512 [==============================] - 2s 238us/step - loss: 0.4520 - val_loss: 0.4623
Epoch 10/20
8512/8512 [==============================] - 2s 236us/step - loss: 0.4512 - val_loss: 0.4613
Epoch 11/20
8512/8512 [==============================] - 2s 239us/step - loss: 0.4509 - val_loss: 0.4611
Epoch 12/20
8512/8512 [==============================] - 2s 236us/step - loss: 0.4501 - val_loss: 0.4620
Epoch 13/20
8512/8512 [==============================] - 2s 238us/step - loss: 0.4498 - val_loss: 0.4608
Epoch 14/20
8512/8512 [==============================] - 2s 237us/step - loss: 0.4490 - val_loss: 0.4619
Epoch 15/20
8512/8512 [==============================] - 2s 237us/step - loss: 0.4493 - val_loss: 0.4619
Epoch 16/20
8512/8512 [==============================] - 2s 239us/step - loss: 0.4492 - val_loss: 0.4615
Epoch 17/20
8512/8512 [==============================] - 2s 252us/step - loss: 0.4483 - val_loss: 0.4625
Epoch 18/20
8512/8512 [==============================] - 2s 240us/step - loss: 0.4483 - val_loss: 0.4605
Epoch 19/20
8512/8512 [==============================] - 2s 237us/step - loss: 0.4486 - val_loss: 0.4610
Epoch 20/20
8512/8512 [==============================] - 2s 238us/step - loss: 0.4477 - val_loss: 0.4602
```



```
8513/8513 [==============================] - 0s 50us/step
normalized test_loss: 0.430043821326682
unnormalized test_loss: 0.1003117102198283
```

```
In [22]: """2c: Bidirectional RNN""" """32"""

         model = Sequential()
         model.add(layers.Bidirectional(layers.LSTM(32)))
         model.add(layers.Dense(32, activation='relu'))
         model.add(layers.Dropout(0.6))
         model.add(layers.Dense(output_size, activation='tanh'))

         """COMPILE YOUR MODEL"""
         model.compile(optimizer=RMSprop(), loss='mae')


         """TRAINING YOUR MODEL"""
         epoch_size = 20
         batch_size = 32
         history = model.fit(training_data,
                             training_labels,
                             epochs=epoch_size,
                             batch_size=batch_size,
                             validation_data = (val_data, val_labels))


         """Plotting results"""
         loss = history.history['loss']
         val_loss = history.history['val_loss']
         epochs = range(1, len(loss) + 1)
         plt.figure()
         plt.plot(epochs, loss, 'bo', label='Training loss')
         plt.plot(epochs, val_loss, 'b', label='Validation loss')
         plt.title('Training and validation losses')
         plt.legend()
         plt.show()


         """PREDICTION - TESTING DATA"""
         test_loss = model.evaluate(test_data, test_labels)
         print('normalized test_loss:', test_loss)
         print('unnormalized test_loss:', test_loss*std[0])

         """Save your model:"""
         model.save('C:/Users/aza8223/OneDrive - University of Tulsa/to_be_transfered/python/Pro
Train on 8512 samples, validate on 8512 samples
Epoch 1/20
8512/8512 [==============================] - 4s 437us/step - loss: 0.5145 - val_loss: 0.4820
Epoch 2/20
8512/8512 [==============================] - 1s 175us/step - loss: 0.4710 - val_loss: 0.4657
Epoch 3/20
8512/8512 [==============================] - 1s 174us/step - loss: 0.4612 - val_loss: 0.4654
Epoch 4/20
8512/8512 [==============================] - 1s 173us/step - loss: 0.4572 - val_loss: 0.4634
Epoch 5/20
8512/8512 [==============================] - 1s 174us/step - loss: 0.4546 - val_loss: 0.4642
Epoch 6/20
8512/8512 [==============================] - 1s 173us/step - loss: 0.4514 - val_loss: 0.4620
Epoch 7/20
8512/8512 [==============================] - 1s 174us/step - loss: 0.4505 - val_loss: 0.4616
Epoch 8/20
8512/8512 [==============================] - 1s 172us/step - loss: 0.4501 - val_loss: 0.4609
Epoch 9/20
8512/8512 [==============================] - 1s 172us/step - loss: 0.4502 - val_loss: 0.4608
Epoch 10/20
8512/8512 [==============================] - 1s 174us/step - loss: 0.4494 - val_loss: 0.4609
Epoch 11/20
8512/8512 [==============================] - 1s 173us/step - loss: 0.4493 - val_loss: 0.4611
Epoch 12/20
8512/8512 [==============================] - 1s 174us/step - loss: 0.4498 - val_loss: 0.4605
Epoch 13/20
8512/8512 [==============================] - 1s 172us/step - loss: 0.4500 - val_loss: 0.4611
Epoch 14/20
8512/8512 [==============================] - 1s 175us/step - loss: 0.4490 - val_loss: 0.4615
Epoch 15/20
8512/8512 [==============================] - 1s 172us/step - loss: 0.4486 - val_loss: 0.4606
Epoch 16/20
8512/8512 [==============================] - 1s 175us/step - loss: 0.4490 - val_loss: 0.4605
Epoch 17/20
8512/8512 [==============================] - 1s 173us/step - loss: 0.4484 - val_loss: 0.4607
Epoch 18/20
8512/8512 [==============================] - 1s 173us/step - loss: 0.4484 - val_loss: 0.4607
Epoch 19/20
8512/8512 [==============================] - 2s 177us/step - loss: 0.4479 - val_loss: 0.4607
Epoch 20/20
8512/8512 [==============================] - 1s 172us/step - loss: 0.4488 - val_loss: 0.4607
```



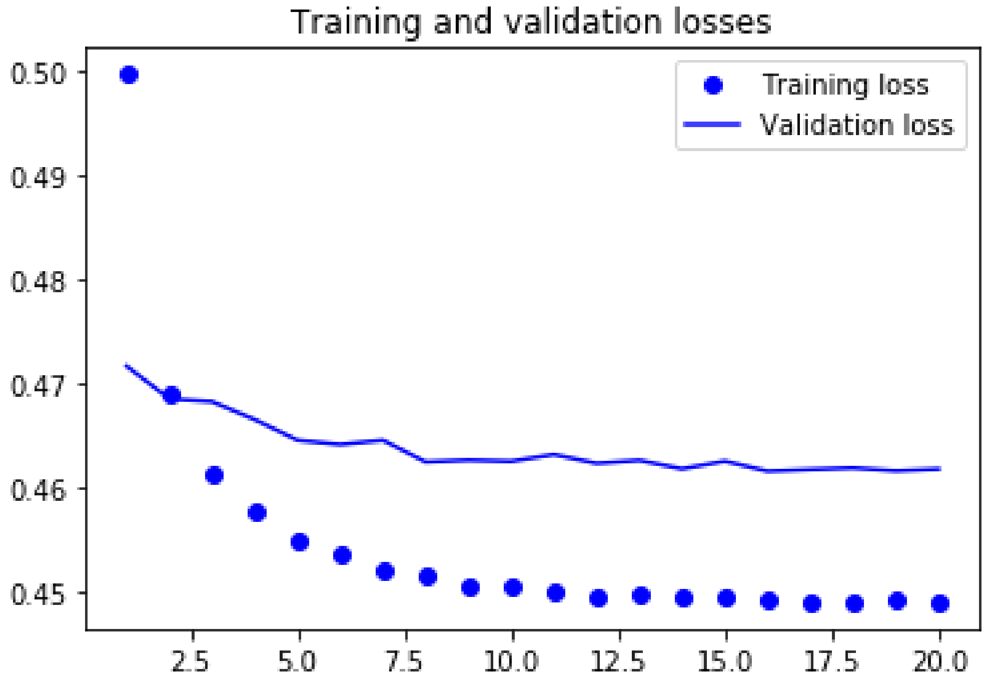
```
8513/8513 [==============================] - 0s 28us/step
normalized test_loss: 0.43051723399394604
unnormalized test_loss: 0.10042213811563316
```

```
In [23]: """2d: Training and evaluating an LSTM using reversed sequences  10.23"""

         """First reverse days (sequentions or times) in your training and validation data,
         but not labels"""
         """tanh seems better choice even for hidden layers"""
         x_train = [x[::-1] for x in training_data] #It will reverse days (times)
         x_test = [x[::-1] for x in test_data]
         x_train = np.asarray(x_train).astype('float32')
         x_test = np.asarray(x_test).astype('float32')

         x_val = [x[::-1] for x in val_data] #It will reverse days (times)
         x_val = np.asarray(x_val).astype('float32')


         model = Sequential()
         model.add(layers.LSTM(32))
         model.add(layers.Dropout(0.5))
         model.add(layers.Dense(output_size, activation='tanh'))

         """COMPILE YOUR MODEL"""
```

```
model.compile(optimizer=RMSprop(), loss='mae')


"""TRAINING YOUR MODEL"""
epoch_size = 20
batch_size = 32
history = model.fit(x_train,
                    training_labels,
                    epochs=epoch_size,
                    batch_size=batch_size,
                    validation_data = (x_val, val_labels))


"""Plotting results"""
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation losses')
plt.legend()
plt.show()


"""PREDICTION - TESTING DATA"""
test_loss = model.evaluate(test_data, test_labels)
print('normalized test_loss:', test_loss)
print('unnormalized test_loss:', test_loss*std[0])

"""Save your model:"""
model.save('C:/Users/aza8223/OneDrive - University of Tulsa/to_be_transfered/python/Pro
```

```
Train on 8512 samples, validate on 8512 samples
Epoch 1/20
8512/8512 [==============================] - 3s 347us/step - loss: 0.4998 - val_loss: 0.4717
Epoch 2/20
8512/8512 [==============================] - 1s 142us/step - loss: 0.4689 - val_loss: 0.4685
Epoch 3/20
8512/8512 [==============================] - 1s 143us/step - loss: 0.4613 - val_loss: 0.4683
Epoch 4/20
8512/8512 [==============================] - 1s 145us/step - loss: 0.4578 - val_loss: 0.4665
Epoch 5/20
8512/8512 [==============================] - 1s 143us/step - loss: 0.4548 - val_loss: 0.4645
Epoch 6/20
8512/8512 [==============================] - 1s 142us/step - loss: 0.4536 - val_loss: 0.4641
Epoch 7/20
8512/8512 [==============================] - 1s 145us/step - loss: 0.4520 - val_loss: 0.4645
```

```
Epoch 8/20
8512/8512 [==============================] - 1s 142us/step - loss: 0.4514 - val_loss: 0.4624
Epoch 9/20
8512/8512 [==============================] - 1s 143us/step - loss: 0.4506 - val_loss: 0.4626
Epoch 10/20
8512/8512 [==============================] - 1s 144us/step - loss: 0.4504 - val_loss: 0.4625
Epoch 11/20
8512/8512 [==============================] - 1s 144us/step - loss: 0.4501 - val_loss: 0.4632
Epoch 12/20
8512/8512 [==============================] - 1s 144us/step - loss: 0.4495 - val_loss: 0.4623
Epoch 13/20
8512/8512 [==============================] - 1s 146us/step - loss: 0.4498 - val_loss: 0.4626
Epoch 14/20
8512/8512 [==============================] - 1s 144us/step - loss: 0.4494 - val_loss: 0.4618
Epoch 15/20
8512/8512 [==============================] - 1s 142us/step - loss: 0.4494 - val_loss: 0.4625
Epoch 16/20
8512/8512 [==============================] - 1s 145us/step - loss: 0.4493 - val_loss: 0.4616
Epoch 17/20
8512/8512 [==============================] - 1s 143us/step - loss: 0.4490 - val_loss: 0.4617
Epoch 18/20
8512/8512 [==============================] - 1s 144us/step - loss: 0.4491 - val_loss: 0.4619
Epoch 19/20
8512/8512 [==============================] - 1s 145us/step - loss: 0.4493 - val_loss: 0.4616
Epoch 20/20
8512/8512 [==============================] - 1s 144us/step - loss: 0.4491 - val_loss: 0.4618
```



```
8513/8513 [==============================] - 0s 30us/step
normalized test_loss: 0.4403145650853971
unnormalized test_loss: 0.10270745646840351
```

```
In [24]: """3a: CONV1 """ """The worst one""" """ good but needs more epoch, but it is fast
         and there was not any overfitting"""
         """I added dropout to get over overfittiing"""
         """Dont use conv1 network if you use 1 day as sequence"""

         if input_size >5:
             model = Sequential()
             model.add(layers.Conv1D(32, input_size-5, activation='relu',
                                     input_shape=(None, training_data.shape[-1])))
             model.add(layers.GlobalMaxPooling1D())#Global maxpooling gives you scalar output
             model.add(layers.Dropout(0.7))
             model.add(layers.Dense(output_size, activation='tanh' ))

             model.summary()

             """COMPILE YOUR MODEL"""
             model.compile(optimizer=RMSprop(), loss='mae')


             """TRAINING YOUR MODEL"""
             epoch_size = 100
             batch_size = 32
             history = model.fit(training_data,
                                 training_labels,
                                 epochs=epoch_size,
                                 batch_size=batch_size,
                                 validation_data = (val_data, val_labels))


             """Plotting results"""
             loss = history.history['loss']
             val_loss = history.history['val_loss']
             epochs = range(1, len(loss) + 1)
             plt.figure()
             plt.plot(epochs, loss, 'bo', label='Training loss')
             plt.plot(epochs, val_loss, 'b', label='Validation loss')
             plt.title('Training and validation losses')
             plt.legend()
             plt.show()
```

```
        """PREDICTION - TESTING DATA"""
        test_loss = model.evaluate(test_data, test_labels)
        print('normalized test_loss:', test_loss)
        print('unnormalized test_loss:', test_loss*std[0])


        """Save your model:"""
        model.save('C:/Users/aza8223/OneDrive - University of Tulsa/to_be_transfered/python
    else:
        print("for 1 day sequence you cannot use Conv layer")
```

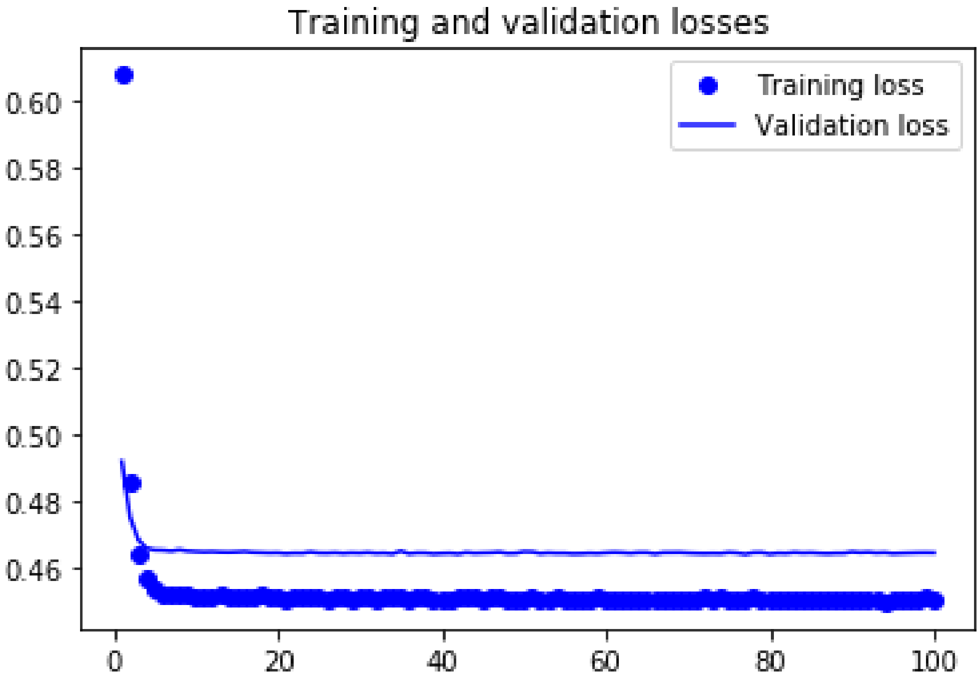
```
-----------------------------------------------------------------
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, None, 32)          224
-----------------------------------------------------------------
global_max_pooling1d_1 (Glob (None, 32)                0
-----------------------------------------------------------------
dropout_19 (Dropout)         (None, 32)                0
-----------------------------------------------------------------
dense_28 (Dense)             (None, 7)                 231
=================================================================
Total params: 455
Trainable params: 455
Non-trainable params: 0
-----------------------------------------------------------------
Train on 8512 samples, validate on 8512 samples
Epoch 1/100
8512/8512 [==============================] - 2s 201us/step - loss: 0.6079 - val_loss: 0.4921
Epoch 2/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4859 - val_loss: 0.4755
Epoch 3/100
8512/8512 [==============================] - 0s 47us/step - loss: 0.4640 - val_loss: 0.4687
Epoch 4/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4569 - val_loss: 0.4662
Epoch 5/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4539 - val_loss: 0.4656
Epoch 6/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4524 - val_loss: 0.4656
Epoch 7/100
8512/8512 [==============================] - 0s 46us/step - loss: 0.4523 - val_loss: 0.4653
Epoch 8/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4528 - val_loss: 0.4657
Epoch 9/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4521 - val_loss: 0.4653
Epoch 10/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4518 - val_loss: 0.4651
```

```
Epoch 11/100
8512/8512 [==============================] - 0s 46us/step - loss: 0.4515 - val_loss: 0.4650
Epoch 12/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4519 - val_loss: 0.4651
Epoch 13/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4521 - val_loss: 0.4650
Epoch 14/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4513 - val_loss: 0.4649
Epoch 15/100
8512/8512 [==============================] - 0s 46us/step - loss: 0.4516 - val_loss: 0.4650
Epoch 16/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4518 - val_loss: 0.4652
Epoch 17/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4513 - val_loss: 0.4649
Epoch 18/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4520 - val_loss: 0.4648
Epoch 19/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4515 - val_loss: 0.4647
Epoch 20/100
8512/8512 [==============================] - 0s 46us/step - loss: 0.4517 - val_loss: 0.4648
Epoch 21/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4511 - val_loss: 0.4645
Epoch 22/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4513 - val_loss: 0.4647
Epoch 23/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4516 - val_loss: 0.4647
Epoch 24/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4516 - val_loss: 0.4650
Epoch 25/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4515 - val_loss: 0.4647
Epoch 26/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4510 - val_loss: 0.4647
Epoch 27/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4517 - val_loss: 0.4648
Epoch 28/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4518 - val_loss: 0.4646
Epoch 29/100
8512/8512 [==============================] - 0s 46us/step - loss: 0.4511 - val_loss: 0.4648
Epoch 30/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4515 - val_loss: 0.4647
Epoch 31/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4516 - val_loss: 0.4649
Epoch 32/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4511 - val_loss: 0.4646
Epoch 33/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4512 - val_loss: 0.4647
Epoch 34/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4518 - val_loss: 0.4644
```

```
Epoch 35/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4515 - val_loss: 0.4653
Epoch 36/100
8512/8512 [==============================] - 0s 46us/step - loss: 0.4510 - val_loss: 0.4645
Epoch 37/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4514 - val_loss: 0.4648
Epoch 38/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4513 - val_loss: 0.4647
Epoch 39/100
8512/8512 [==============================] - 0s 43us/step - loss: 0.4511 - val_loss: 0.4644
Epoch 40/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4507 - val_loss: 0.4646
Epoch 41/100
8512/8512 [==============================] - 0s 46us/step - loss: 0.4510 - val_loss: 0.4647
Epoch 42/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4512 - val_loss: 0.4644
Epoch 43/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4512 - val_loss: 0.4649
Epoch 44/100
8512/8512 [==============================] - 0s 46us/step - loss: 0.4513 - val_loss: 0.4647
Epoch 45/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4507 - val_loss: 0.4648
Epoch 46/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4518 - val_loss: 0.4648
Epoch 47/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4517 - val_loss: 0.4650
Epoch 48/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4510 - val_loss: 0.4646
Epoch 49/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4510 - val_loss: 0.4646
Epoch 50/100
8512/8512 [==============================] - 0s 46us/step - loss: 0.4510 - val_loss: 0.4651
Epoch 51/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4513 - val_loss: 0.4651
Epoch 52/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4505 - val_loss: 0.4646
Epoch 53/100
8512/8512 [==============================] - 0s 46us/step - loss: 0.4510 - val_loss: 0.4648
Epoch 54/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4513 - val_loss: 0.4647
Epoch 55/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4506 - val_loss: 0.4647
Epoch 56/100
8512/8512 [==============================] - 0s 46us/step - loss: 0.4508 - val_loss: 0.4645
Epoch 57/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4509 - val_loss: 0.4645
Epoch 58/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4511 - val_loss: 0.4647
```

```
Epoch 59/100
8512/8512 [==============================] - 0s 46us/step - loss: 0.4513 - val_loss: 0.4646
Epoch 60/100
8512/8512 [==============================] - 0s 46us/step - loss: 0.4509 - val_loss: 0.4648
Epoch 61/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4511 - val_loss: 0.4649
Epoch 62/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4508 - val_loss: 0.4648
Epoch 63/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4506 - val_loss: 0.4646
Epoch 64/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4509 - val_loss: 0.4649
Epoch 65/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4505 - val_loss: 0.4649
Epoch 66/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4511 - val_loss: 0.4645
Epoch 67/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4511 - val_loss: 0.4648
Epoch 68/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4510 - val_loss: 0.4649
Epoch 69/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4510 - val_loss: 0.4649
Epoch 70/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4509 - val_loss: 0.4647
Epoch 71/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4506 - val_loss: 0.4646
Epoch 72/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4514 - val_loss: 0.4646
Epoch 73/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4509 - val_loss: 0.4646
Epoch 74/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4514 - val_loss: 0.4646
Epoch 75/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4510 - val_loss: 0.4649
Epoch 76/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4511 - val_loss: 0.4647
Epoch 77/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4505 - val_loss: 0.4643
Epoch 78/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4516 - val_loss: 0.4648
Epoch 79/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4509 - val_loss: 0.4649
Epoch 80/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4511 - val_loss: 0.4644
Epoch 81/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4510 - val_loss: 0.4647
Epoch 82/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4508 - val_loss: 0.4646
```

```
Epoch 83/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4511 - val_loss: 0.4649
Epoch 84/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4510 - val_loss: 0.4647
Epoch 85/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4507 - val_loss: 0.4648
Epoch 86/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4506 - val_loss: 0.4646
Epoch 87/100
8512/8512 [==============================] - 0s 46us/step - loss: 0.4508 - val_loss: 0.4645
Epoch 88/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4505 - val_loss: 0.4647
Epoch 89/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4510 - val_loss: 0.4647
Epoch 90/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4506 - val_loss: 0.4651
Epoch 91/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4505 - val_loss: 0.4649
Epoch 92/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4506 - val_loss: 0.4650
Epoch 93/100
8512/8512 [==============================] - 0s 46us/step - loss: 0.4508 - val_loss: 0.4647
Epoch 94/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4501 - val_loss: 0.4648
Epoch 95/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4505 - val_loss: 0.4646
Epoch 96/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4508 - val_loss: 0.4647
Epoch 97/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4508 - val_loss: 0.4647
Epoch 98/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4508 - val_loss: 0.4648
Epoch 99/100
8512/8512 [==============================] - 0s 45us/step - loss: 0.4513 - val_loss: 0.4648
Epoch 100/100
8512/8512 [==============================] - 0s 44us/step - loss: 0.4510 - val_loss: 0.4648
```



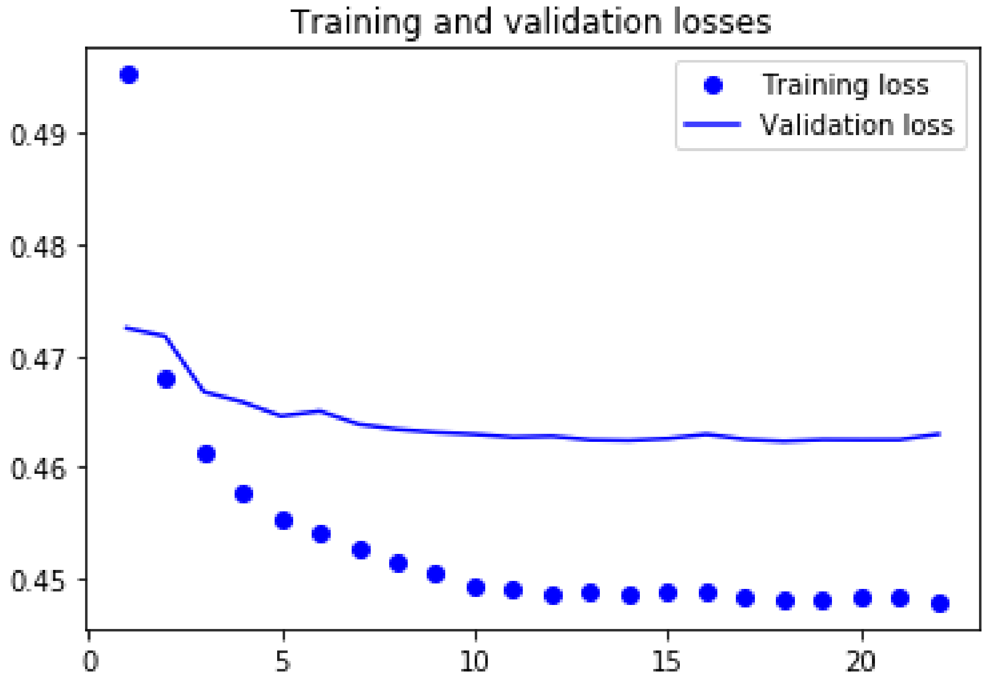
```
8513/8513 [==============================] - 0s 15us/step
normalized test_loss: 0.4353888113514196
unnormalized test_loss: 0.10155847872085057
```

```
In [25]: """3b: Combining CNNs and RNNs to process long sequences"""
         """ not bad and it is fast"""


         if input_size >5:
             model = Sequential()
             model.add(layers.Conv1D(32, input_size-5, activation='relu',
                                     input_shape=(None, training_data.shape[-1])))
             model.add(layers.MaxPooling1D(3))
             model.add(layers.GRU(32, dropout=0.2, recurrent_dropout=0.2))
             model.add(layers.Dropout(0.4))
             model.add(layers.Dense(output_size, activation='tanh'))

             """COMPILE YOUR MODEL"""
             model.compile(optimizer=RMSprop(), loss='mae')


             """TRAINING YOUR MODEL"""
             epoch_size = 22
             batch_size = 32
```

```
            history = model.fit(training_data,
                                training_labels,
                                epochs=epoch_size,
                                batch_size=batch_size,
                                validation_data = (val_data, val_labels))


            """Plotting results"""
            loss = history.history['loss']
            val_loss = history.history['val_loss']
            epochs = range(1, len(loss) + 1)
            plt.figure()
            plt.plot(epochs, loss, 'bo', label='Training loss')
            plt.plot(epochs, val_loss, 'b', label='Validation loss')
            plt.title('Training and validation losses')
            plt.legend()
            plt.show()

            """PREDICTION - TESTING DATA"""
            test_loss = model.evaluate(test_data, test_labels)
            print('normalized test_loss:', test_loss)
            print('unnormalized test_loss:', test_loss*std[0])

            """Save your model:"""
            model.save('C:/Users/aza8223/OneDrive - University of Tulsa/to_be_transfered/python
        else:
            print("for 1 day sequence you cannot use Conv layer")
```

```
Train on 8512 samples, validate on 8512 samples
Epoch 1/22
8512/8512 [==============================] - 3s 307us/step - loss: 0.4953 - val_loss: 0.4725
Epoch 2/22
8512/8512 [==============================] - 1s 82us/step - loss: 0.4681 - val_loss: 0.4717
Epoch 3/22
8512/8512 [==============================] - 1s 86us/step - loss: 0.4612 - val_loss: 0.4668
Epoch 4/22
8512/8512 [==============================] - 1s 82us/step - loss: 0.4577 - val_loss: 0.4658
Epoch 5/22
8512/8512 [==============================] - 1s 82us/step - loss: 0.4553 - val_loss: 0.4646
Epoch 6/22
8512/8512 [==============================] - 1s 84us/step - loss: 0.4541 - val_loss: 0.4650
Epoch 7/22
8512/8512 [==============================] - 1s 83us/step - loss: 0.4526 - val_loss: 0.4638
Epoch 8/22
8512/8512 [==============================] - 1s 84us/step - loss: 0.4514 - val_loss: 0.4634
Epoch 9/22
8512/8512 [==============================] - 1s 82us/step - loss: 0.4504 - val_loss: 0.4631
```

```
Epoch 10/22
8512/8512 [==============================] - 1s 84us/step - loss: 0.4494 - val_loss: 0.4629
Epoch 11/22
8512/8512 [==============================] - 1s 84us/step - loss: 0.4490 - val_loss: 0.4627
Epoch 12/22
8512/8512 [==============================] - 1s 83us/step - loss: 0.4486 - val_loss: 0.4628
Epoch 13/22
8512/8512 [==============================] - 1s 84us/step - loss: 0.4489 - val_loss: 0.4624
Epoch 14/22
8512/8512 [==============================] - 1s 83us/step - loss: 0.4485 - val_loss: 0.4624
Epoch 15/22
8512/8512 [==============================] - 1s 84us/step - loss: 0.4487 - val_loss: 0.4625
Epoch 16/22
8512/8512 [==============================] - 1s 83us/step - loss: 0.4488 - val_loss: 0.4629
Epoch 17/22
8512/8512 [==============================] - 1s 81us/step - loss: 0.4483 - val_loss: 0.4625
Epoch 18/22
8512/8512 [==============================] - 1s 82us/step - loss: 0.4480 - val_loss: 0.4623
Epoch 19/22
8512/8512 [==============================] - 1s 84us/step - loss: 0.4480 - val_loss: 0.4625
Epoch 20/22
8512/8512 [==============================] - 1s 84us/step - loss: 0.4482 - val_loss: 0.4624
Epoch 21/22
8512/8512 [==============================] - 1s 82us/step - loss: 0.4482 - val_loss: 0.4624
Epoch 22/22
8512/8512 [==============================] - 1s 81us/step - loss: 0.4478 - val_loss: 0.4630
```



```
8513/8513 [==============================] - 0s 24us/step
normalized test_loss: 0.43392875382271284
unnormalized test_loss: 0.10121790675943491
```

```
In [26]: """4: Using DAG network"""
         """When I used different layer types I put here the best architecture and
         diagram for my prediction"""
         """One input but Multiple output. Diagram is shown in the report"""

         from keras import layers
         from keras import Input
         from keras.models import Model



         """Input layer:"""
         inputt = Input(shape=(input_size,training_data.shape[-1]), dtype='float32', name='previ
         a = layers.GRU(32, dropout=0.2, recurrent_dropout=0.2, activation='relu')(inputt)
         a = layers.Dropout(0.4)(a)

         """Output layers for each day:"""
         x = layers.Dense(32, activation='relu')(a)
         x = layers.Dropout(0.4)(x)
         day_1 = layers.Dense(1,activation='tanh', name='day1')(x)

         y = layers.Dense(32, activation='relu')(a)
         y = layers.Dropout(0.4)(y)
         day_2 = layers.Dense(1,activation='tanh', name='day2')(y)

         z = layers.Dense(32, activation='relu')(a)
         z = layers.Dropout(0.4)(z)
         day_3 = layers.Dense(1,activation='tanh', name='day3')(z)

         v = layers.Dense(32, activation='relu')(a)
         v = layers.Dropout(0.4)(v)
         day_4 = layers.Dense(1,activation='tanh', name='day4')(v)

         w = layers.Dense(32, activation='relu')(a)
         w = layers.Dropout(0.4)(w)
         day_5 = layers.Dense(1,activation='tanh', name='day5')(w)

         b = layers.Dense(32, activation='relu')(a)
         b = layers.Dropout(0.4)(b)
         day_6 = layers.Dense(1,activation='tanh', name='day6')(b)
```

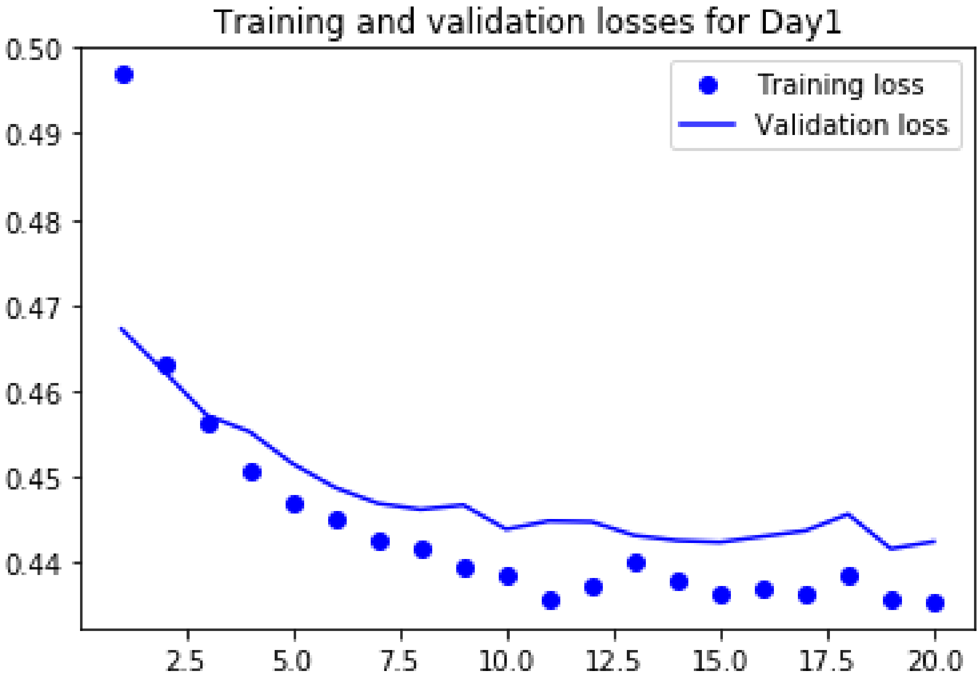
```
c = layers.Dense(32, activation='relu')(a)
c = layers.Dropout(0.4)(c)
day_7 = layers.Dense(1,activation='tanh', name='day7')(c)


"""Fully connected API model:"""
model = Model(inputt, [day_1, day_2, day_3, day_4, day_5, day_6, day_7])

"""Compiling:"""
"""I could add multiple losses but my problem isa regression so only loss here is mae""
"""I can also define different loss weights for different outputs, but that would be
good to use it when we have different type of loss functions. Just in case I have
used different weights but it didnt affaect my results much"""

model.compile(optimizer=RMSprop(), loss='mae')


"""TRAINING YOUR MODEL. Here I will assign target labels for each days seperately"""
epoch_size = 20
batch_size = 32
history = model.fit(training_data,
                   [training_labels[:,0],
                    training_labels[:,1],
                    training_labels[:,2],
                    training_labels[:,3],
                    training_labels[:,4],
                    training_labels[:,5],
                    training_labels[:,6]],
                    epochs=epoch_size,
                    batch_size=batch_size,
                    validation_data = (val_data,
                   [val_labels[:,0],
                    val_labels[:,1],
                    val_labels[:,2],
                    val_labels[:,3],
                    val_labels[:,4],
                    val_labels[:,5],
                    val_labels[:,6]]))



"""Plot losses for each day in different plots"""
"""Predict losses for each day seperately:"""

"""Day1:"""
loss = history.history['day1_loss']
val_loss = history.history['val_day1_loss']
```

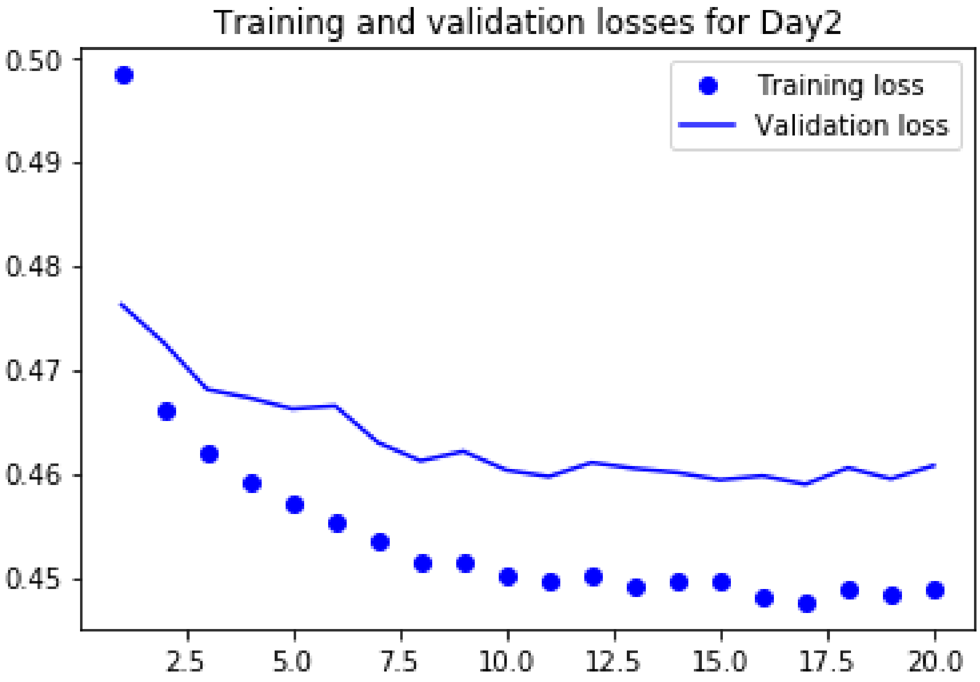
```
epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation losses for Day1')
plt.legend()
plt.show()


###################################################
"""Day2:"""
loss = history.history['day2_loss']
val_loss = history.history['val_day2_loss']
epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation losses for Day2')
plt.legend()
plt.show()


###################################################
"""Day3:"""
loss = history.history['day3_loss']
val_loss = history.history['val_day3_loss']
epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation losses for Day3')
plt.legend()
plt.show()



###################################################
"""Day4:"""
loss = history.history['day4_loss']
val_loss = history.history['val_day4_loss']
epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation losses for Day4')
plt.legend()
plt.show()
```

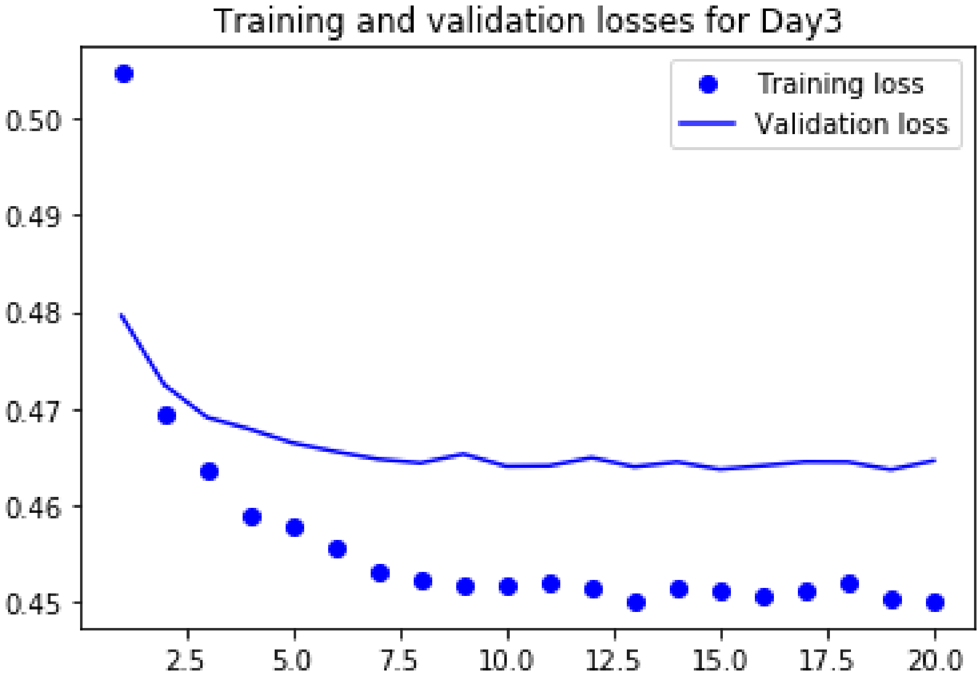
```
###################################################
"""Day5:"""
loss = history.history['day5_loss']
val_loss = history.history['val_day5_loss']
epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation losses for Day5')
plt.legend()
plt.show()


###################################################
"""Day6:"""
loss = history.history['day6_loss']
val_loss = history.history['val_day6_loss']
epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation losses for Day6')
plt.legend()
plt.show()


###################################################
"""Day7:"""
loss = history.history['day7_loss']
val_loss = history.history['val_day7_loss']
epochs = range(1, len(loss) + 1)
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation losses for Day7')
plt.legend()
plt.show()


###################################################
###################################################
"""PREDICTION - TESTING DATA for each days both normalized and unnormalized
for DAG model"""
test_LossAndAcc = model.evaluate(test_data, [i for i in np.transpose(test_labels)])
test_losses = test_LossAndAcc[1:]
for i, k in enumerate(test_losses):
    print('normalized test_loss of ', 'day', i+1, 'is', test_losses[i])
    print('unnormalized test_loss of ', 'day', i+1, 'is', test_losses[i]*std[0])


###################################################
###################################################
"""Base case for each day and mean of mae:"""
preds = np.mean(val_data[:, :, 0], axis=1)
day = np.zeros((val_labels.shape[1], val_labels.shape[0]))
mae_base1 = np.zeros((val_labels.shape[1],))
for i,j in enumerate(np.transpose(val_labels)):
    day[i] = val_labels[:,i]
    mae_base1[i] = np.nanmean(np.abs(preds - day[i]))
    print('normalized MAE of base model for day ', i+1, " is ", mae_base1[i])
    print('unnormalized MAE of base model for day ', i+1, " is ", mae_base1[i]*std[0])
mae_base_mean = mae_base1.mean()
print('mean of normalized MAE of base model of week ', " is ", mae_base_mean)
print('mean of unnormalized MAE of base model of week ', " is ", mae_base_mean*std[0])

"""Save your model:"""
model.save('C:/Users/aza8223/OneDrive - University of Tulsa/to_be_transfered/python/Pro
```

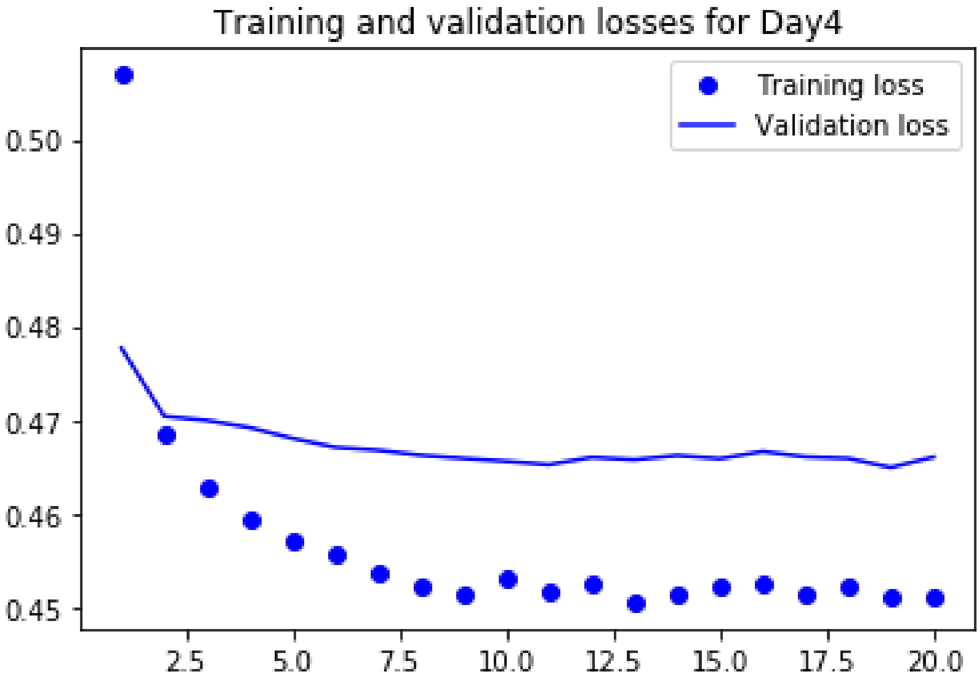
```
Train on 8512 samples, validate on 8512 samples
Epoch 1/20
8512/8512 [==============================] - 4s 524us/step - loss: 3.5102 - day1_loss: 0.4969 -
Epoch 2/20
8512/8512 [==============================] - 1s 166us/step - loss: 3.2741 - day1_loss: 0.4633 -
Epoch 3/20
8512/8512 [==============================] - 1s 165us/step - loss: 3.2376 - day1_loss: 0.4564 -
Epoch 4/20
8512/8512 [==============================] - 1s 167us/step - loss: 3.2095 - day1_loss: 0.4506 -
Epoch 5/20
8512/8512 [==============================] - 1s 168us/step - loss: 3.1966 - day1_loss: 0.4469 -
Epoch 6/20
8512/8512 [==============================] - 1s 166us/step - loss: 3.1831 - day1_loss: 0.4451 -
Epoch 7/20
8512/8512 [==============================] - 1s 167us/step - loss: 3.1716 - day1_loss: 0.4428 -
Epoch 8/20
8512/8512 [==============================] - 1s 169us/step - loss: 3.1636 - day1_loss: 0.4417 -
Epoch 9/20
8512/8512 [==============================] - 1s 166us/step - loss: 3.1576 - day1_loss: 0.4394 -
Epoch 10/20
8512/8512 [==============================] - 1s 170us/step - loss: 3.1579 - day1_loss: 0.4386 -
Epoch 11/20
8512/8512 [==============================] - 1s 168us/step - loss: 3.1490 - day1_loss: 0.4357 -
Epoch 12/20
8512/8512 [==============================] - 1s 167us/step - loss: 3.1539 - day1_loss: 0.4374 -
Epoch 13/20
8512/8512 [==============================] - 1s 168us/step - loss: 3.1493 - day1_loss: 0.4401 -
```

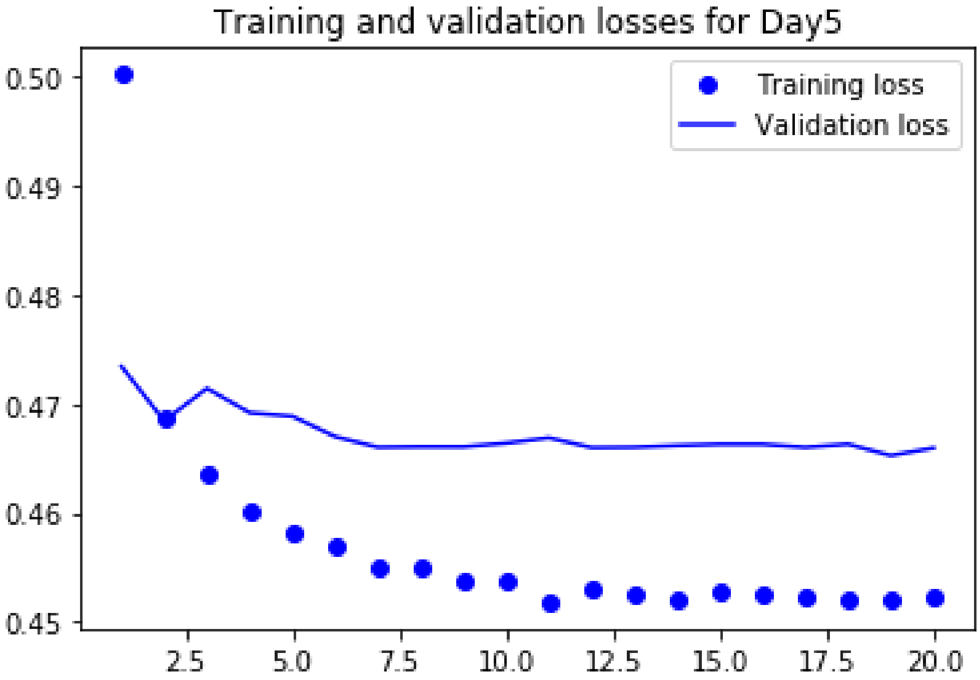
```
Epoch 14/20
8512/8512 [==============================] - 1s 170us/step - loss: 3.1488 - day1_loss: 0.4378 -
Epoch 15/20
8512/8512 [==============================] - 1s 170us/step - loss: 3.1496 - day1_loss: 0.4365 -
Epoch 16/20
8512/8512 [==============================] - 1s 171us/step - loss: 3.1477 - day1_loss: 0.4371 -
Epoch 17/20
8512/8512 [==============================] - 1s 171us/step - loss: 3.1462 - day1_loss: 0.4363 -
Epoch 18/20
8512/8512 [==============================] - 1s 170us/step - loss: 3.1506 - day1_loss: 0.4385 -
Epoch 19/20
8512/8512 [==============================] - 1s 173us/step - loss: 3.1449 - day1_loss: 0.4359 -
Epoch 20/20
8512/8512 [==============================] - 1s 173us/step - loss: 3.1447 - day1_loss: 0.4355 -
```

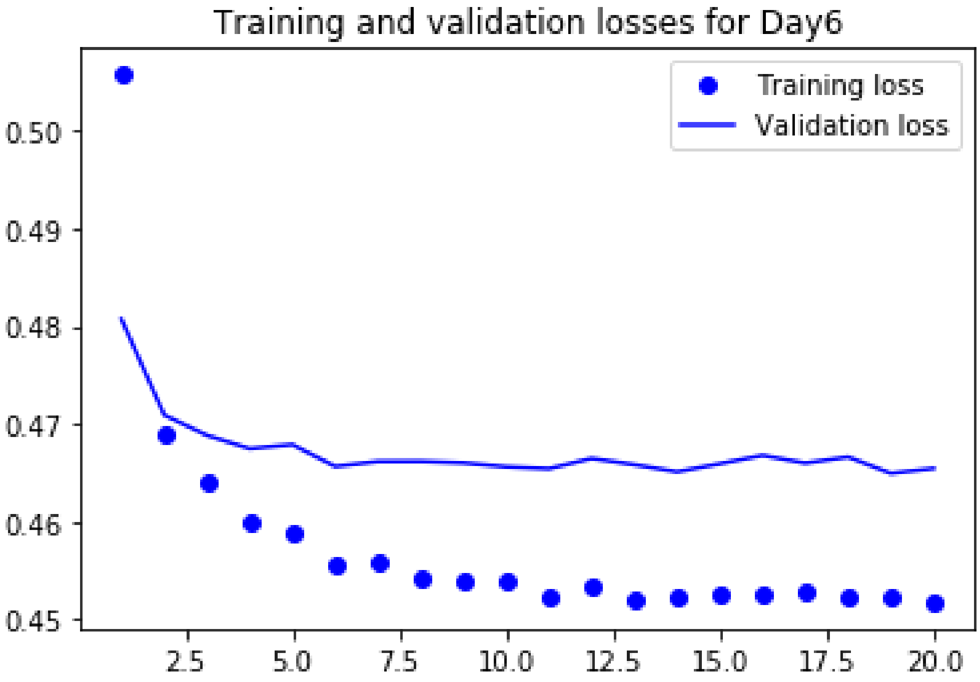


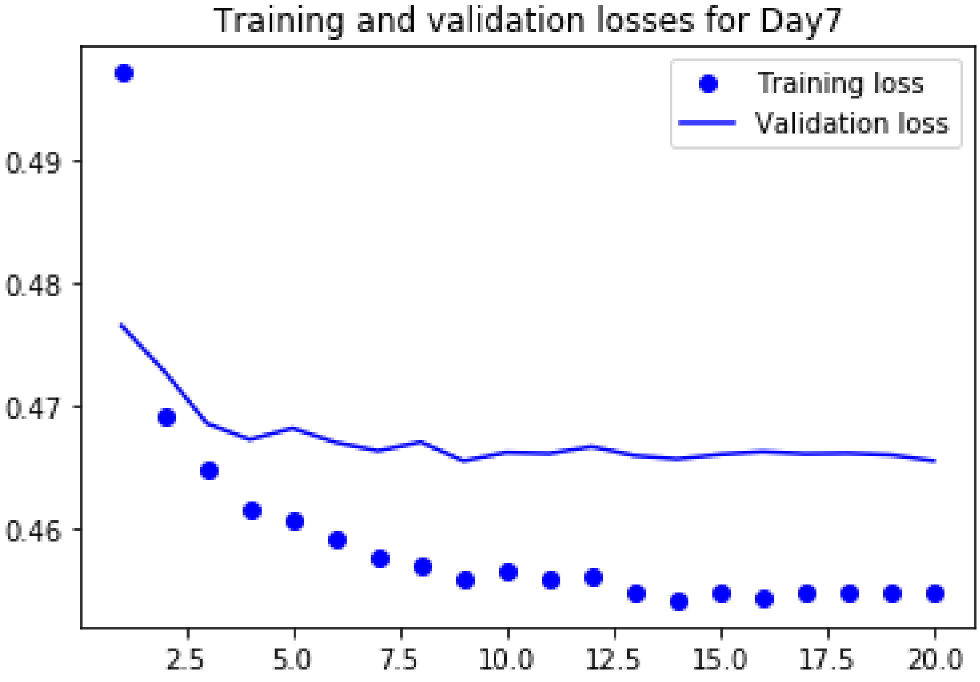

Training and validation losses for Day2
Training loss
Validation loss
0.50
0.49
0.48
0.47
0.46
0.45
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Training and validation losses for Day3
Training loss
Validation loss
0.50
0.49
0.48
0.47
0.46
0.45
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Training and validation losses for Day4
● Training loss
Validation loss
0.50
0.49
0.48
0.47
0.46
0.45
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Training and validation losses for Day5
● Training loss
Validation loss
0.50
0.49
0.48
0.47
0.46
0.45
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0





```
8513/8513 [==============================] - 0s 37us/step
normalized test_loss of  day 1 is 0.4106062191795493
```

```
unnormalized test_loss of  day 1 is 0.09577770922444996
normalized test_loss of  day 2 is 0.4297662665053343
unnormalized test_loss of  day 2 is 0.10024696798327377
normalized test_loss of  day 3 is 0.43311141378644574
unnormalized test_loss of  day 3 is 0.10102725461469265
normalized test_loss of  day 4 is 0.43553364854168136
unnormalized test_loss of  day 4 is 0.10159226333892454
normalized test_loss of  day 5 is 0.43697615299874043
unnormalized test_loss of  day 5 is 0.10192874088356388
normalized test_loss of  day 6 is 0.43737908359365374
unnormalized test_loss of  day 6 is 0.10202272818223254
normalized test_loss of  day 7 is 0.4347035212388235
unnormalized test_loss of  day 7 is 0.10139862844564097
normalized MAE of base model for day  1  is  0.5893916490332757
unnormalized MAE of base model for day  1  is  0.13748106907202862
normalized MAE of base model for day  2  is  0.6158605915959213
unnormalized MAE of base model for day  2  is  0.14365519543891442
normalized MAE of base model for day  3  is  0.6282546054119308
unnormalized MAE of base model for day  3  is  0.146546214122863
normalized MAE of base model for day  4  is  0.6354590758521732
unnormalized MAE of base model for day  4  is  0.14822672367851578
normalized MAE of base model for day  5  is  0.6384783752504914
unnormalized MAE of base model for day  5  is  0.14893100326885297
normalized MAE of base model for day  6  is  0.6422891159124167
unnormalized MAE of base model for day  6  is  0.14981989387498396
normalized MAE of base model for day  7  is  0.646235350870764
unnormalized MAE of base model for day  7  is  0.15074038978254028
mean of normalized MAE of base model of week   is  0.6279955377038533
mean of unnormalized MAE of base model of week   is  0.146485784176957
```