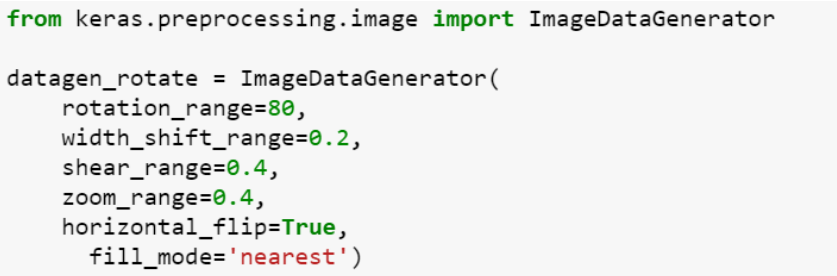
NAME: ANAND KRISHNAMOORTHY
CSCI S-89 INTRODUCTION TO DEEP LEARNING
ASSIGNMENT 4

Problem 1 (5 points)

Please perform image augmentation on an image of a dog of your choice. Produce one modified image for every of the following options: rotation_range, width_shift, shear_range, zoom_range, and horizontal_flip in ImageDataGenerator().

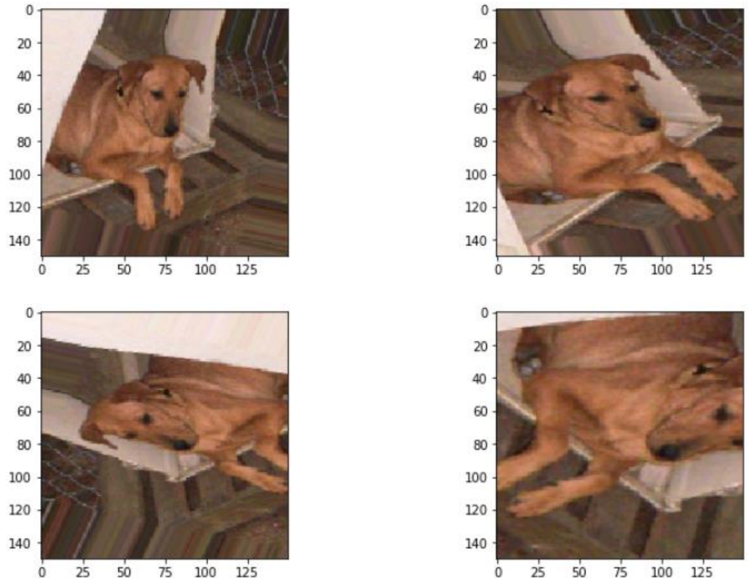
Hint: You may find examples of image augmentation in
5.2-Training_a_convnet_from_scratch_on_a_small_dataset.ipynb notebook useful.

SOLUTION:

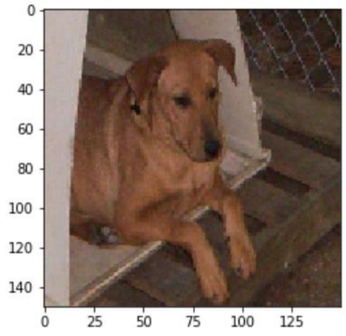


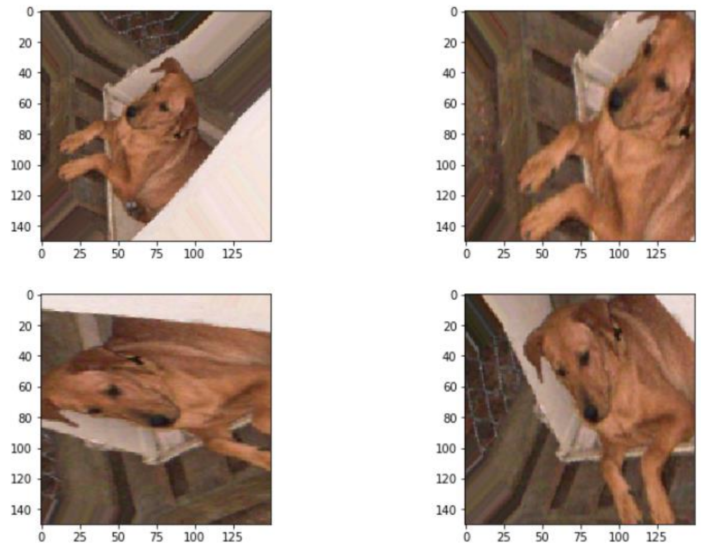
```
from keras.preprocessing.image import ImageDataGenerator

datagen_rotate = ImageDataGenerator(
    rotation_range=80,
    width_shift_range=0.2,
    shear_range=0.4,
    zoom_range=0.4,
    horizontal_flip=True,
      fill_mode='nearest')
```

Augmented Image



Original Image

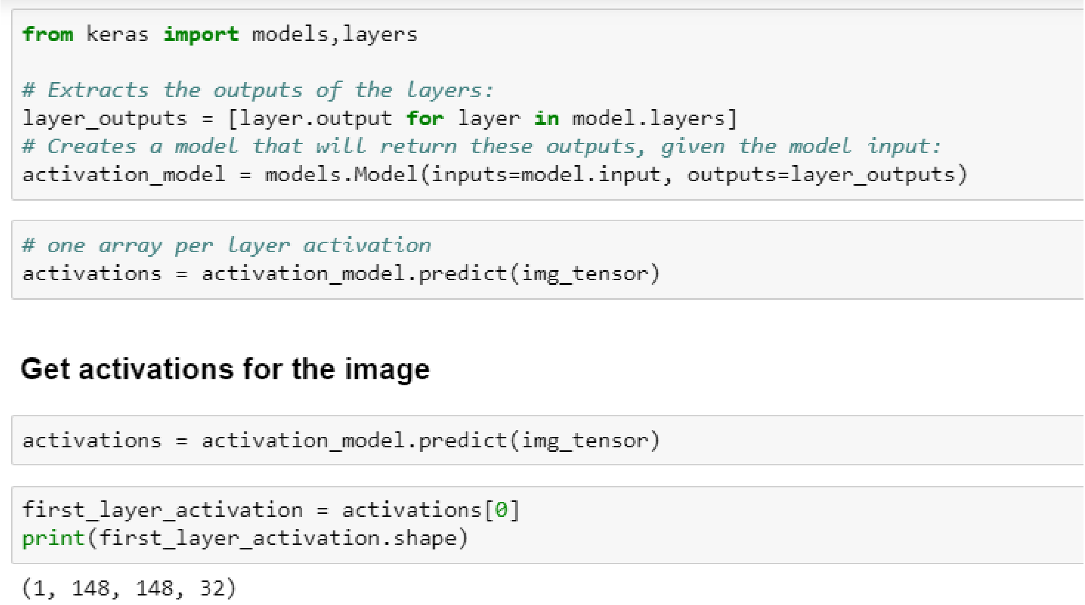




Problem 2 (25 points)

Please consider the Convolutional Neural Network for cat/dog classification saved in cats_and_dogs_small_2.h5. Load the pre-trained network using load_model() – please find an example in 5.4-Visualizing_what_convnets_learn.ipynb. Pick an image of a lion, trim it to size 150x150 pixels (for example, using OpenCV), and then repeat the analysis in the notebook. Lion is a kind of a cat so the results are expected to be close. Please visualize intermediate activations and in the first convnet layer, for the channels with indexes 3 and 30. Also, please select two other convolutional layers and one of MaxPooling layer and generate activation images for those layers.

SOLUTION:



```
from keras import models,layers

# Extracts the outputs of the layers:
layer_outputs = [layer.output for layer in model.layers]
# Creates a model that will return these outputs, given the model input:
activation_model = models.Model(inputs=model.input, outputs=layer_outputs)
```

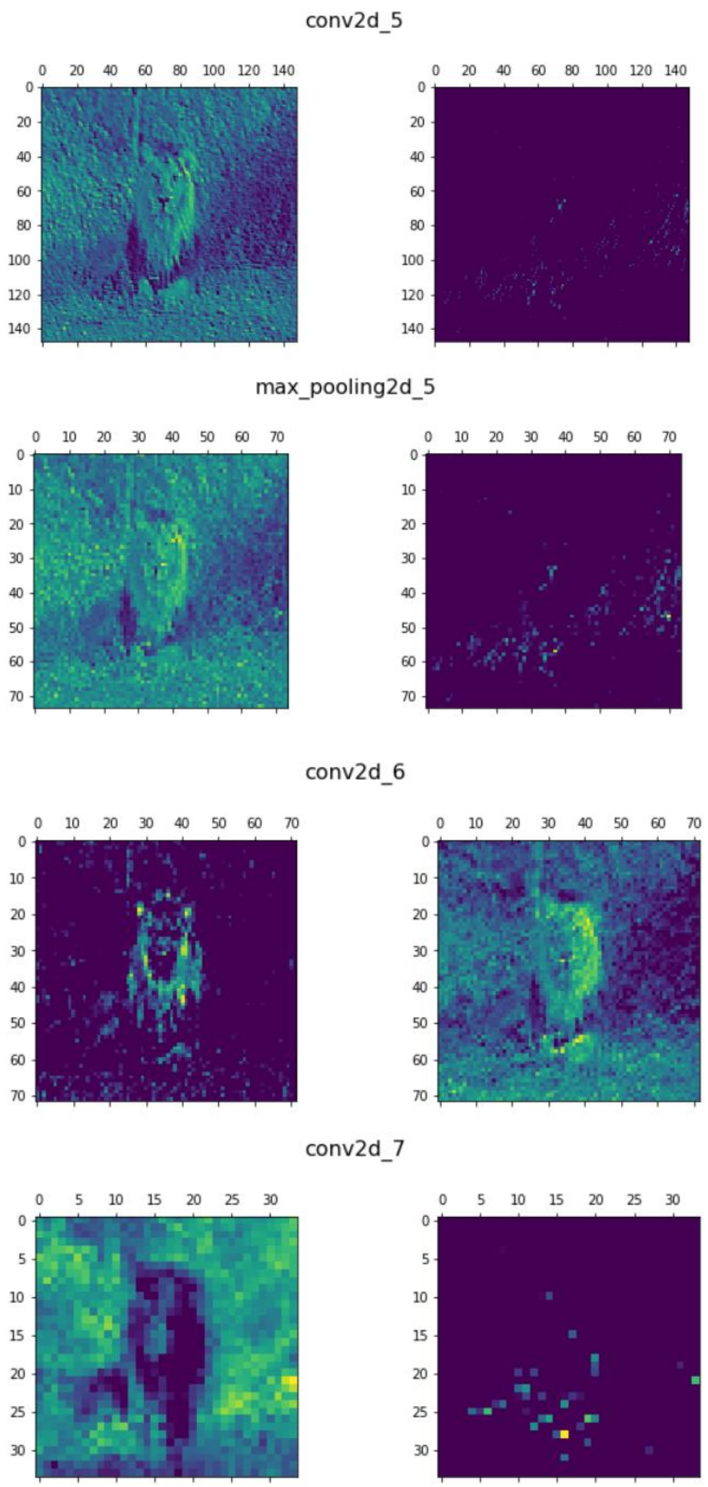
```
# one array per layer activation
activations = activation_model.predict(img_tensor)
```

Get activations for the image

```
activations = activation_model.predict(img_tensor)
```

```
first_layer_activation = activations[0]
print(first_layer_activation.shape)
```

```
(1, 148, 148, 32)
```


conv2d_5

max_pooling2d_5

conv2d_6

conv2d_7

Problem 3 (35 points)

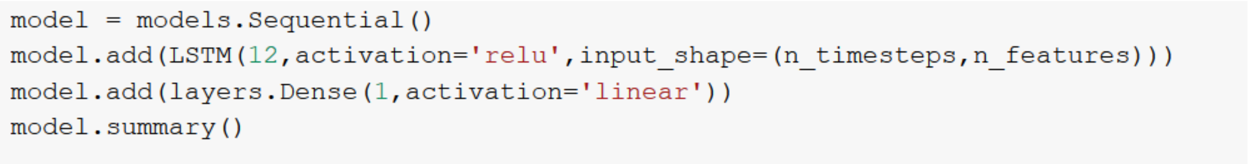
In this problem, we consider observations of temperature (in Celsius) measured every 10 minutes in Jena, Germany between January 2009 and January 2017. The dataset is available at https://www.bgc-jena.mpg.de/wetter/.

Please load the time series:

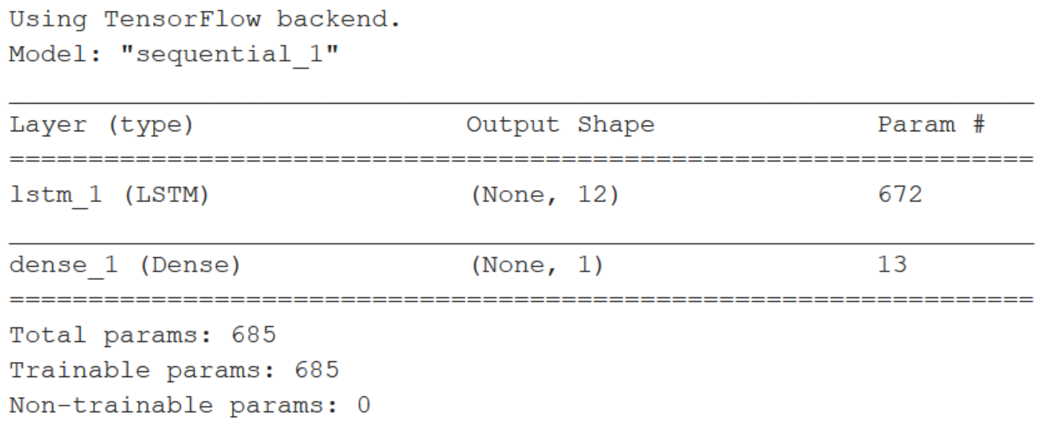
df = pd.read_csv('jena_climate_2009_2016.csv', parse_dates=True, index_col='Date Time')
xt = df['T (degC)']
xt = xt.reset_index(drop=True)

and then break it into train (first 80% of observations) and validation (last 20% of observations) sets – please do not shuffle. Build a Recurrent Neural Network (RNN) of your choice for one step prediction. Please plot the last 1440 observations of temperature (10 days) along with the corresponding one-step ahead predictions.

SOLUTION:

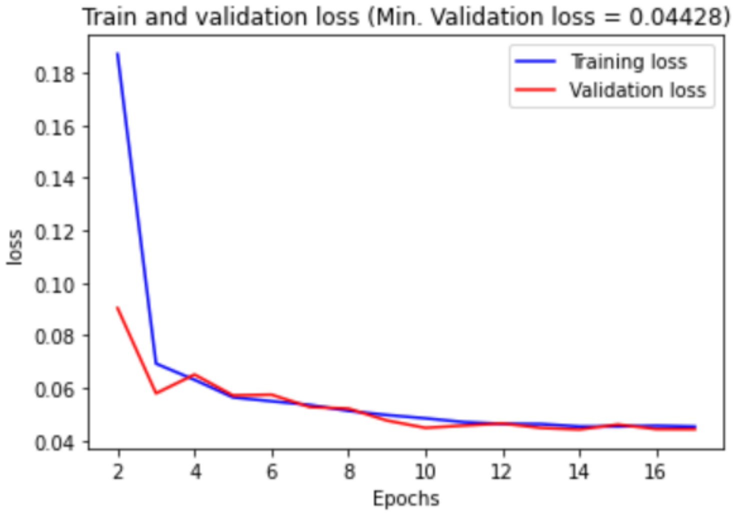


```
model = models.Sequential()
model.add(LSTM(12,activation='relu',input_shape=(n_timesteps,n_features)))
model.add(layers.Dense(1,activation='linear'))
model.summary()
```

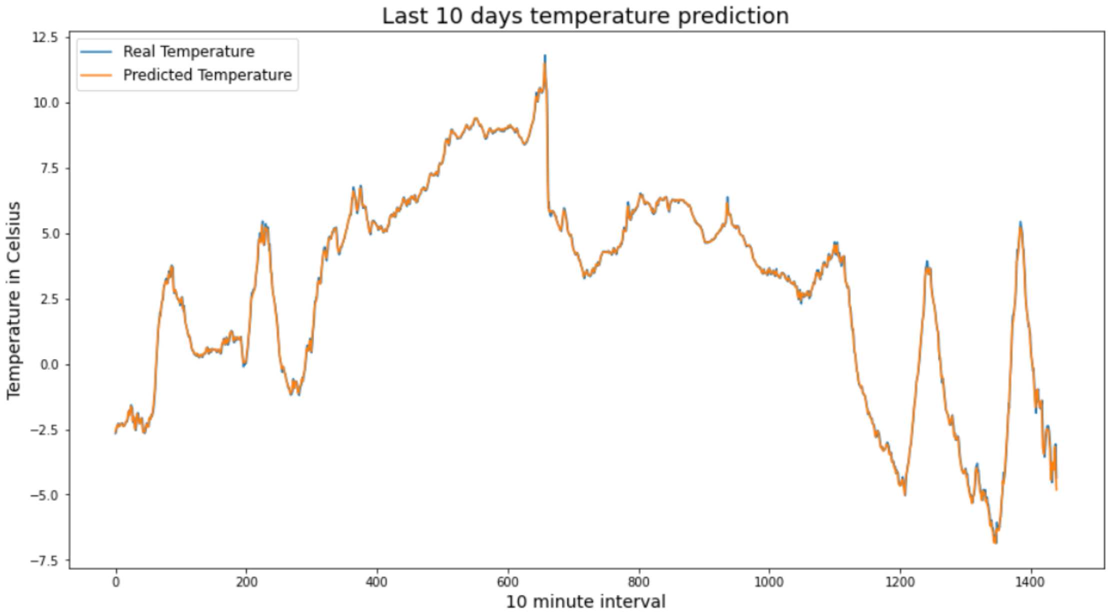


```
Using TensorFlow backend.
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_1 (LSTM)                (None, 12)                672       
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 13        
=================================================================
Total params: 685
Trainable params: 685
Non-trainable params: 0
```

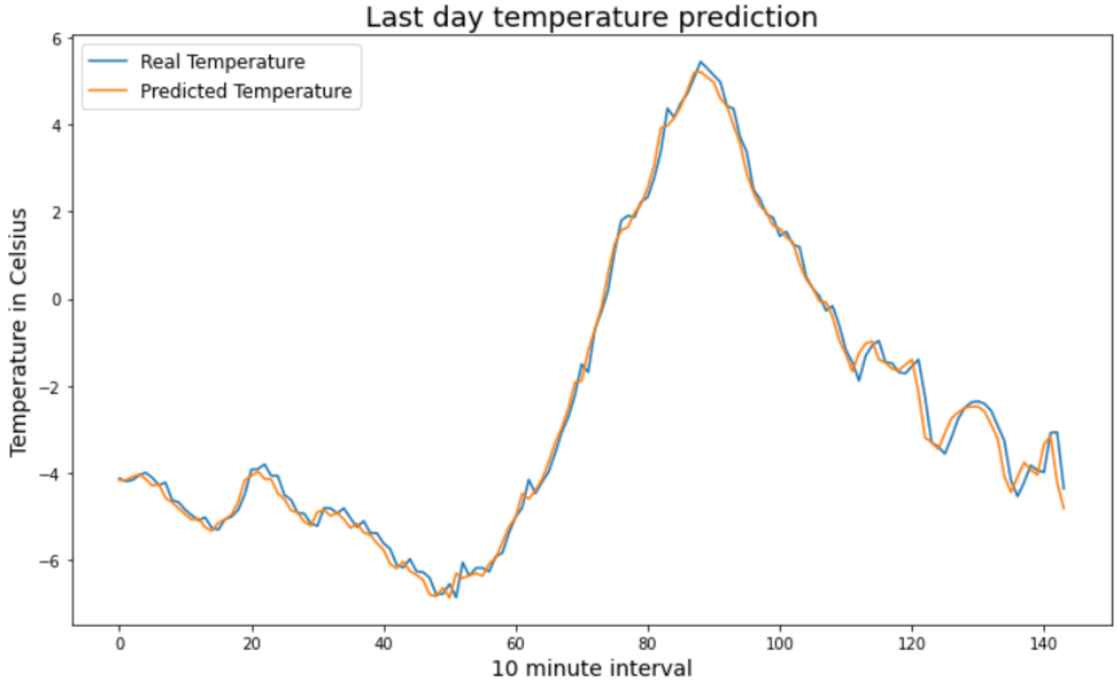
Plotting the Train and Validation Loss



Plotting 10-day temperatures using one step prediction (using the trained model)



Plotting 1-day temperatures using one step prediction (using the trained model)



From the graph we observe that the predicted temperatures are very close to the actual temperatures.