# CSCI S-89, Summer 2022

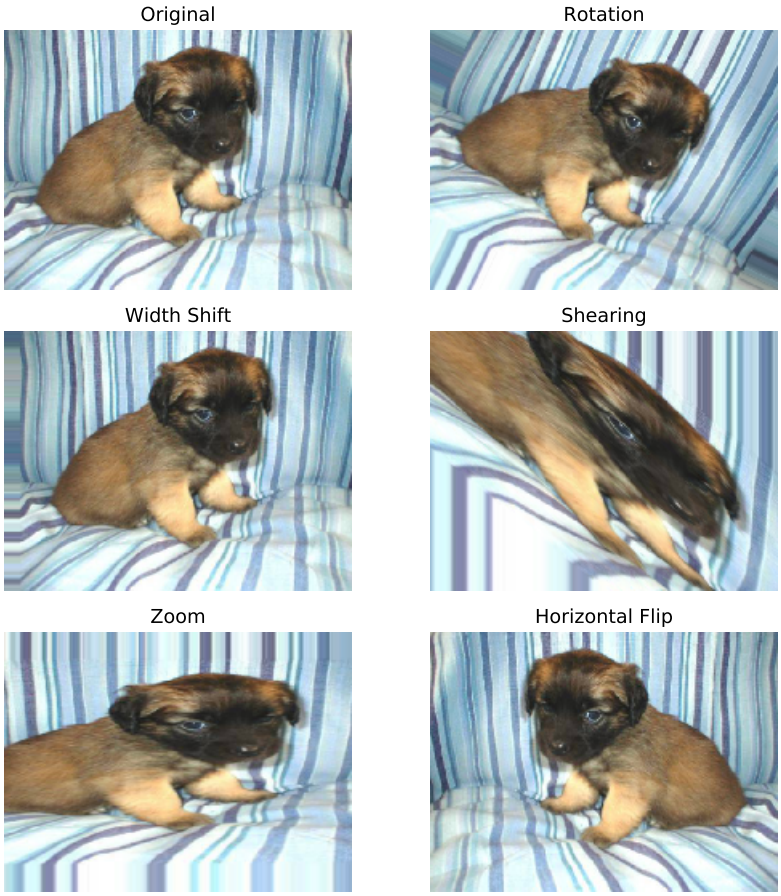
Name:

DUE: July 22, 2022

RE: Assignment #

## 1 Problem 1

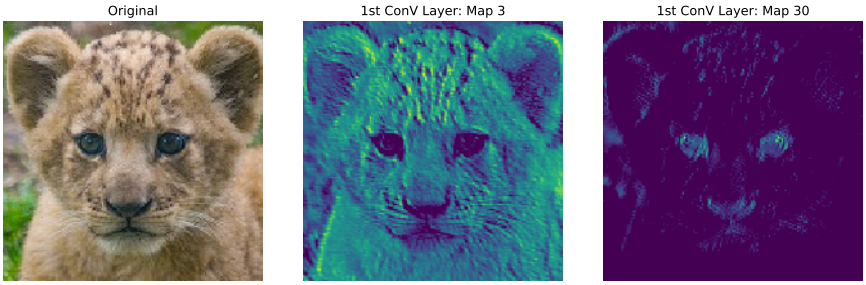
Following the given example, I use the function `ImageDataGenerator` to generate augmented images of a dog. See Appendix Section A for code.




## 2 Problem 2

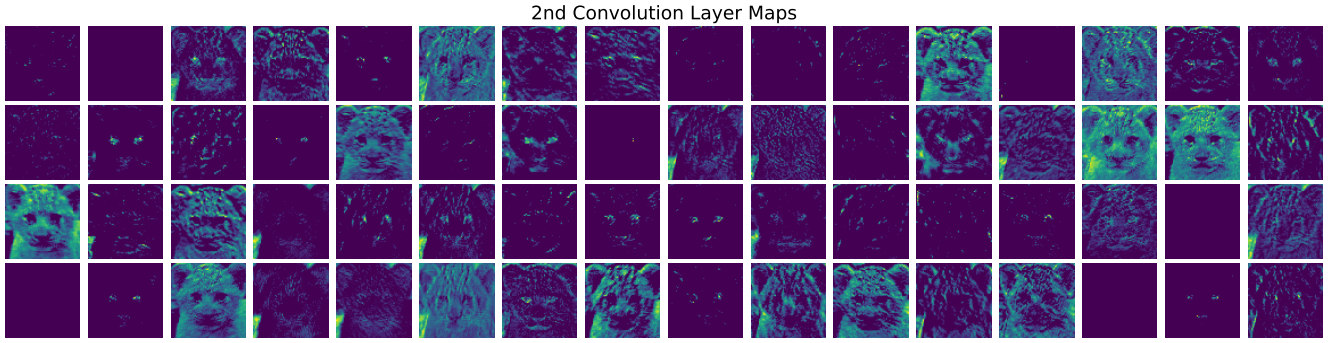
Following the given example, I use an image of cub and feed it as an input to the pre-trained model. The following images show the original one and the channel No.3 and No.30 from the first

convolution layer. See Appendix Section B for code.

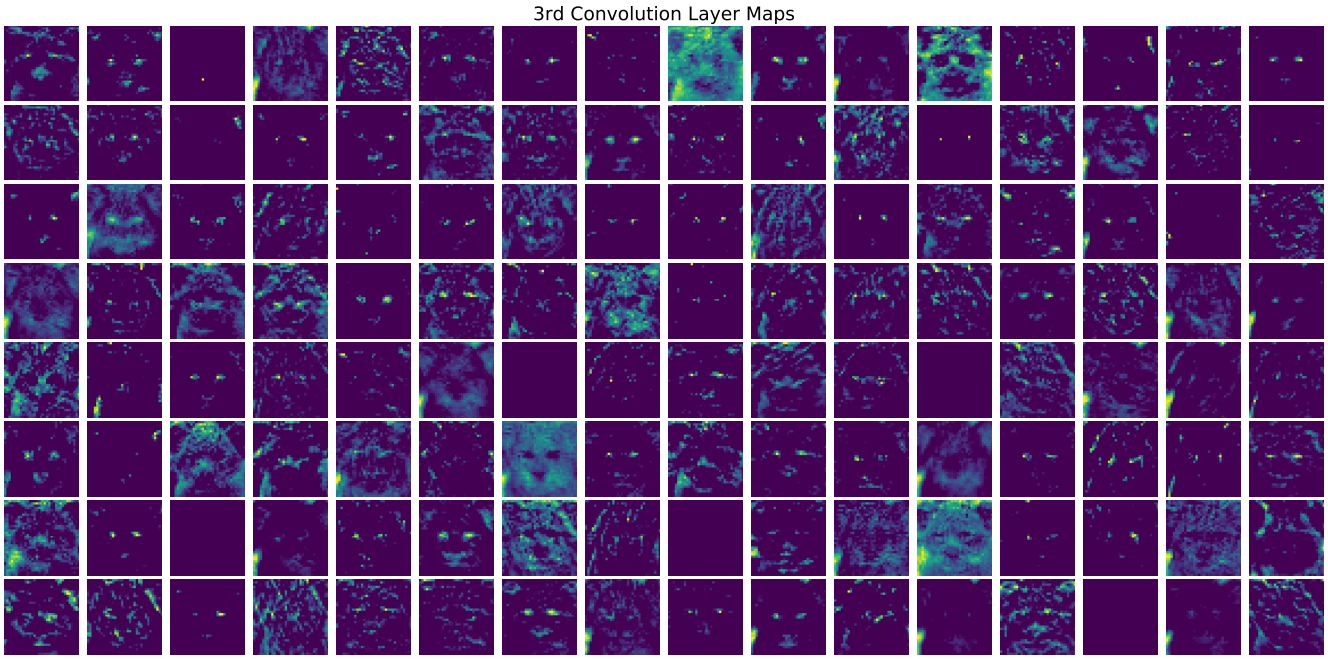



### Activation Images: 2 other Conv Layers

The following images are all 64 feature maps from the second convolution layer.

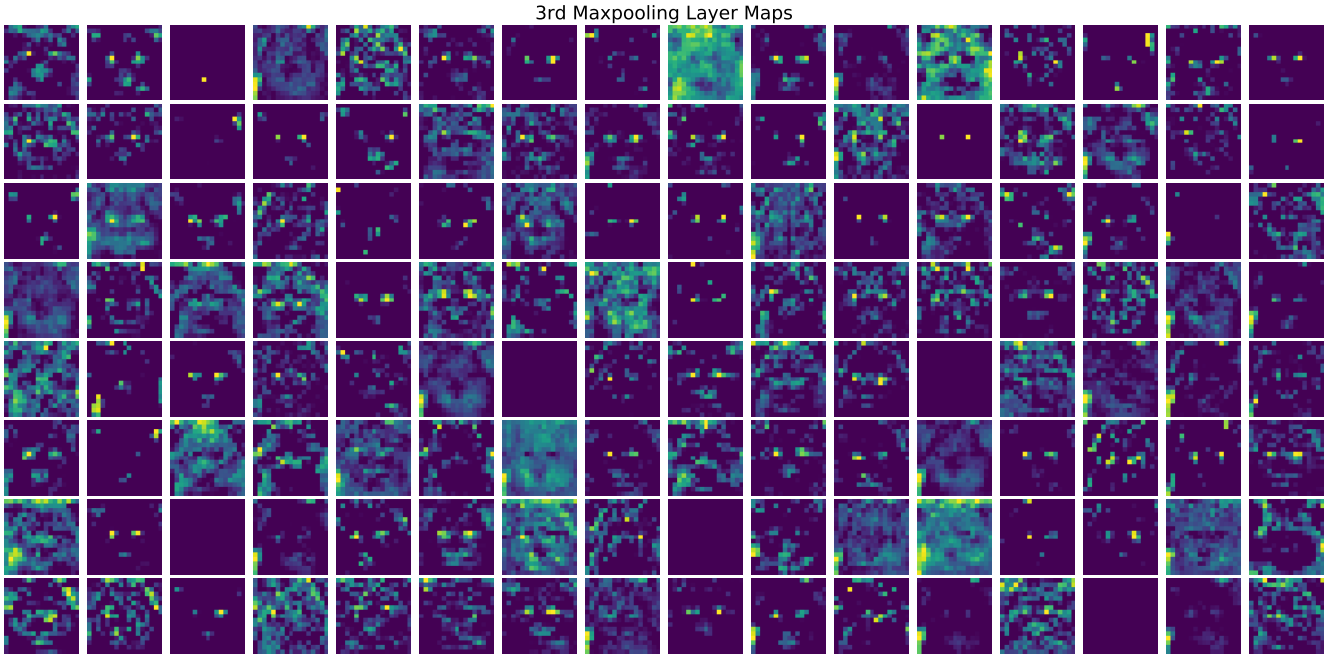



The following images are all 128 feature maps from the third convolution layer.



### Activation Images: One MaxPooling Layer

The following images are all 128 feature maps from the third MaxPooling layer.

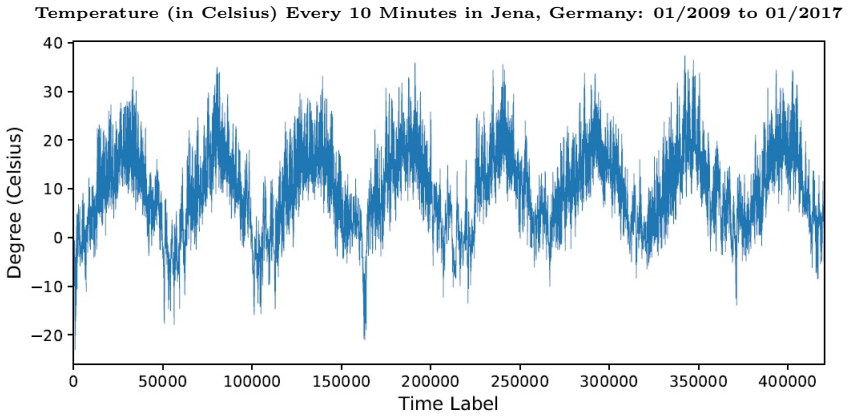



### Discussion

The first convolution layer learns basic shapes and some feature maps still look very similar to the original image. As the model goes deeper, the second and third convolution layers pull more abstract characteristics. The third MaxPooling layer further strengthens these abstract details, such as eyes, nose and ears, etc.

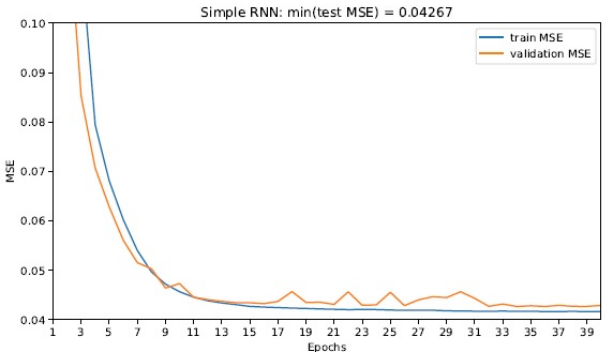
## 3 Problem 3

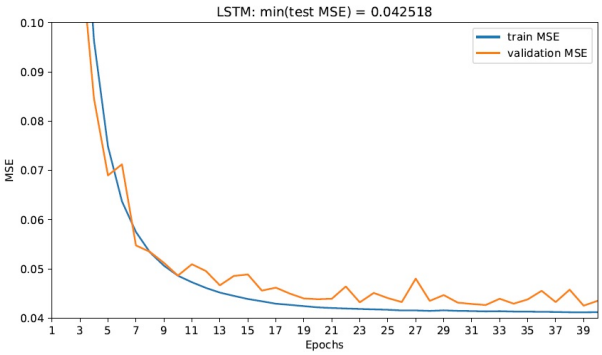
The following chart displays the time series of the temperature data. It has strong annual seasonality but no obvious trend. Notice that there are about 52.5k observations per year.



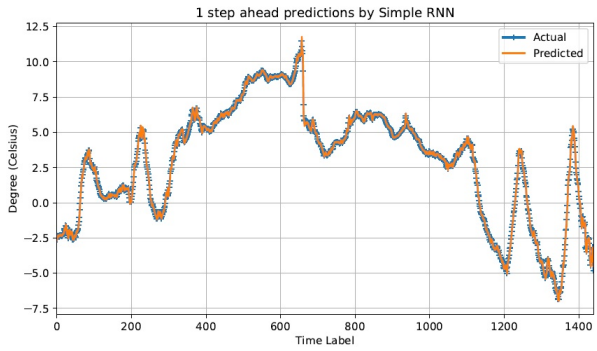
### 3.1 Model and Training

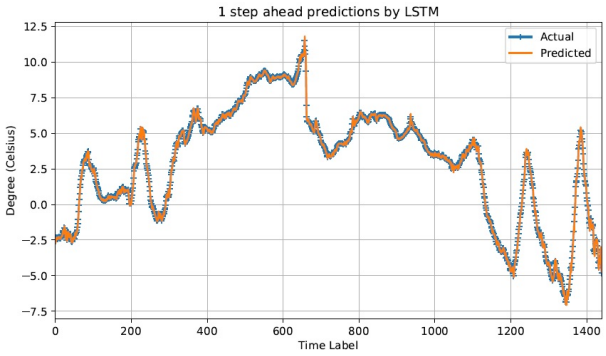
I build a simple RNN model and a LSTM model to make one step forward prediction. The following charts display the training history of the two models. After about 20 epochs both models achieve a validation MSE below 0.05. See Appendix Section C for some code.




The following charts display the actual and predicted temperature for the last 1440 observations. It is not surprising that both models perform very well: the time series data has strong seasonality. With the validation MSE about 0.0426, the rMSE is only 0.21 degree.



# A Problem 1 Code

```
# Load an image
img = image.load_img('./data/dog.478.jpg')
img = image.img_to_array(img)/255.

# Data augmentation
datagen = [ImageDataGenerator(rotation_range=40),
           ImageDataGenerator(width_shift_range=0.3),
           ImageDataGenerator(shear_range=60),
           ImageDataGenerator(zoom_range=0.5),
           ImageDataGenerator(horizontal_flip=True)]

titles = ['Rotation', 'Width Shift', 'Shearing', 'Zoom', 'Horizontal Flip']

fig, axes = plt.subplots(3,2,figsize=(8,8))
ax = axes.flatten()
ax[0].imshow(img)
ax[0].set_title('Original')
ax[0].axis('off');

# Reshape
x = img.reshape((1,) + img.shape)

for i in range(5):

    # Plot
    for batch in datagen[i].flow(x, batch_size=1):
        ax[i+1].imshow(image.array_to_img(batch[0]))
        ax[i+1].set_title(titles[i])
        ax[i+1].axis('off');
        break

fig.tight_layout(h_pad=1, w_pad=1)
plt.savefig('p1_dog.pdf')
```

# B Problem 2 Code

## B.1 Get Layers

```
# Extracts the outputs of the top 8 layers:
layer_outputs = [layer.output for layer in model.layers[:8]]

# Creates a model that will return these outputs, given the model input:
activation_model = models.Model(inputs=model.input, outputs=layer_outputs)
```

```
activations = activation_model.predict(img_tensor)

# Get First conv layer
first_layer_activation = activations[0]

# Get second conv layer
second_layer_activation = activations[2]
print('2nd ConV Layer shape: ', second_layer_activation.shape)

# Get third conv layer
third_layer_activation = activations[4]
print('3rd ConV Layer shape: ', third_layer_activation.shape)

# Get third MaxPooling layer
third_mp_activation = activations[5]

# Plot
fig, axes = plt.subplots(1, 3, figsize=(12,4), dpi=60)

# Original
axes[0].imshow(img_tensor[0])

# Two feature maps
axes[1].imshow(first_layer_activation[0,:,:,3],  cmap='viridis')
axes[2].imshow(first_layer_activation[0,:,:,30], cmap='viridis')

titles = ['Original', '1st ConV Layer: Map 3', '1st ConV Layer: Map 30' ]
for i in range(3):
    axes[i].axis('off')
    axes[i].set_title(titles[i])

plt.tight_layout(pad=2)
plt.savefig('p2_1stCV.pdf')
```

### B.2 3rd MaxPooling Layer

```
# Plot third MaxPooling layer
fig, axes = plt.subplots(8, 16, figsize=(20,10), dpi=60)
ax = axes.flatten()

for i in range(128):
    ax[i].imshow(third_mp_activation[0,:,:,i],  cmap='viridis')
    ax[i].axis('off')
```

```
fig.tight_layout(h_pad=0.5, w_pad=0.5)
fig.subplots_adjust(top=0.95)
fig.suptitle('3rd Maxpooling Layer Maps', fontsize=20)
plt.savefig('p2_3rdMP.pdf')
```

# C Problem 3 Code

## C.1 Simple RNN

```
# Build model
n_features = 1

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

model.summary()

# Compile and train
nepochs = 40

opti = optimizers.Adam(lr=0.0001)

model.compile(optimizer=opti,
              loss='mse')

history1 = model.fit(features_set_train, labels_train,
          epochs=nepochs,
          batch_size=128,
          validation_data=(features_set_test, labels_test))
```

## C.2 LSTM

```
# Build model
n_features = 1

model_lstm = models.Sequential()
model_lstm.add(layers.LSTM(10, return_sequences=False, activation='relu',
               input_shape=(n_timesteps,n_features)))
model_lstm.add(layers.Dense(1, activation='linear'))

model_lstm.summary()

nepochs = 40
```

```
opti = optimizers.Adam(lr=0.0001)

model_lstm.compile(optimizer=opti,
              loss='mse')

history_lstm = model_lstm.fit(features_set_train, labels_train,
          epochs=nepochs,
          batch_size=128,
          validation_data=(features_set_test, labels_test))
```

### C.3 Two Functions for Plots

```
# Two functions

def plot_history(history, title, fname, ylim):
    df = pd.DataFrame(history.history)
    df['Epochs'] = range(1, df.shape[0]+1)
    df.columns = ['train MSE', 'validation MSE', 'Epochs']

    df.plot(x='Epochs', figsize=(9,5))
    plt.title(title+': min(test MSE) = ' + str(round(df['validation MSE'].min(),6)),fontsize=12);
    plt.ylabel('MSE');
    plt.xticks(np.arange(1,df.shape[0]+1, 2.0))
    plt.ylim(ylim);
    plt.savefig(fname)


def save_pred(df_test, title, fname):
    df_test.plot(style=['+-', '-'], figsize=(9,5), ms=5)
    plt.title('1 step ahead predictions by '+ title,fontsize=12)
    plt.grid(True)
    plt.xlabel('Time Label')
    plt.ylabel('Degree (Celsius)');

    plt.savefig(fname)
```