Introduction

Convolution layers are basically collections of filters. Now you are given some gray-scale images and their filtered counterpart, and you are required to find out the filters that generated those images.

Get Started

To help you get a quick start, we will walk you through a workflow using a warm-up example. We will use a very simple CNN to do the filtering: a CNN with only one convolution layer and nothing else. Using Keras, we define the network as follows. You may want to read Keras' document about convolution layers: https://keras.io/layers/convolutional/.

We replaced some parts of the code with ??? and ..., it is your job to read keras' documentation and figure out what happened here.

```
import numpy as np
np.random.seed(0)
from tensorflow import set_random_seed
set_random_seed(0)

import keras
from keras.models import Model
from keras.layers import Input, Conv2D
from matplotlib import pyplot as plt
from PIL import Image
plt.set_cmap('gray')

# The input image is a 224x224x1 grayscale image
a = Input(shape=(224,224,1))
# Read Keras' document about Conv2D:
b = Conv2D(...)(a)
# Combine things together
model = Model(...)
```

```
Using TensorFlow backend.
```

Now we load the images from examples.pkl:

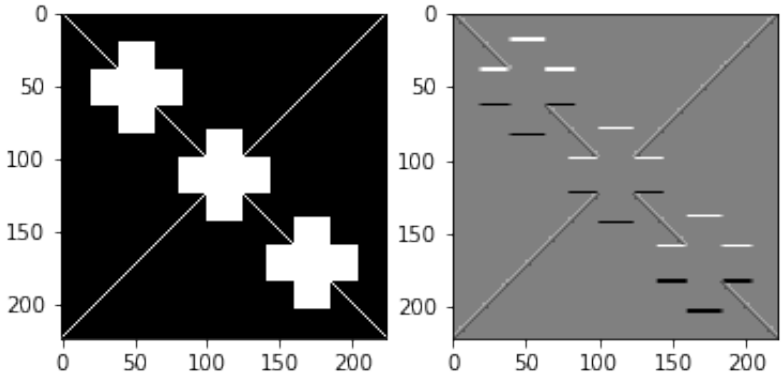
```
import pickle as pk
with open('example.pkl', 'rb') as f:
    example = pk.load(f)
```

Have a look at the image and the filtered image:

```
image = example['img']
image_f = example['img_f']

plt.subplot(121)
plt.imshow(image)
plt.subplot(122)
plt.imshow(image_f)
```

```
<matplotlib.image.AxesImage at 0x11f59feb8>
```



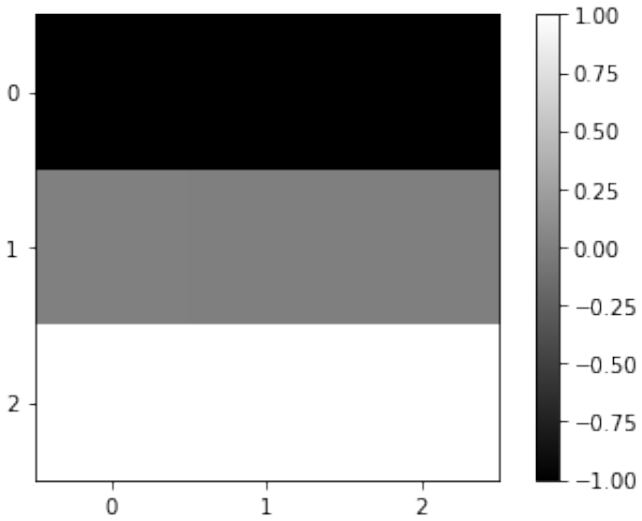
Let's now find out what the filter is. Suppose x is the original image, y is the filtered image, we use SGD to find out filter weights:

```
from keras import optimizers, losses
# read documentation on Keras' optimizers
sgd = optimizers.???(...)
# read documentation on how a Keras model is compiled
model.compile(...)
# expand dimension of input data to make it of shape BxHxWx1,
# B is the batchsize, in our case it's 1.
x = np.expand_dims(np.expand_dims(image, 0), 3)
y = np.expand_dims(np.expand_dims(image_f, 0), 3)
model.fit(x=x, y=y, epochs=5000, verbose=0)
```

```
<keras.callbacks.History at 0x11f5f3cf8>
```

Retrieve the weights from model, and plot the results. It's very clear that this is a horizontal edge detector!

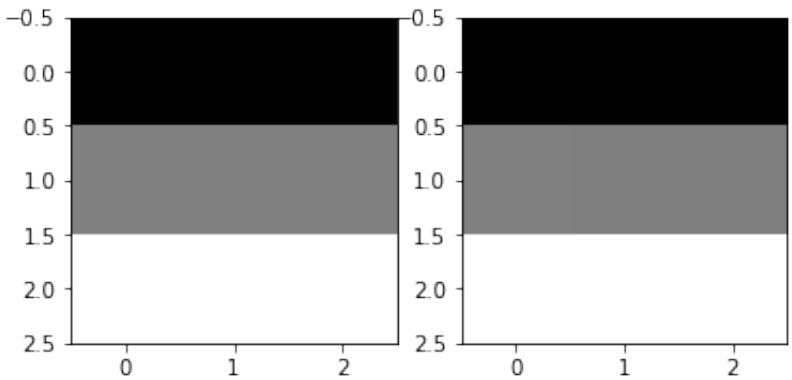
```
conv_weights = model.get_layer(index=1).get_weights()[0]
learned_filter = conv_weights[:,:,0,0]
fig, ax = plt.subplots()
plt.xticks([0,1,2])
plt.yticks([0,1,2])
cs = plt.imshow(learned_filter)
cbar = plt.colorbar(cs)
plt.show()
```



We can check how close this learned filter is to the real filter:

```
# you won't have real_filter in the problem set
real_filter = example['filter'][:,:,0,0]
err = np.linalg.norm(real_filter - learned_filter) / (real_filter.shape[0]
** 2)
print('MSE between real filter and learned filter is {:.6f}'.format(err))
plt.subplot(121)
plt.imshow(real_filter)
plt.subplot(122)
plt.imshow(learned_filter)
```

```
MSE between real filter and learned filter is 0.000372
```



Fair enough! However, note that you won't always get good results like this.

The Problem Set

Now it's your time to code. In this folder we've provided you with a file named `problems.pkl`. This file contains a Dictionary named "problems", which has keys "img" and "img_f". You may load the problem set using the following code:

```
import pickle as pk
with open('problems.pkl', 'rb') as f:
    problems = pk.load(f)
```

The properties of the unknown filters are provided as follows:

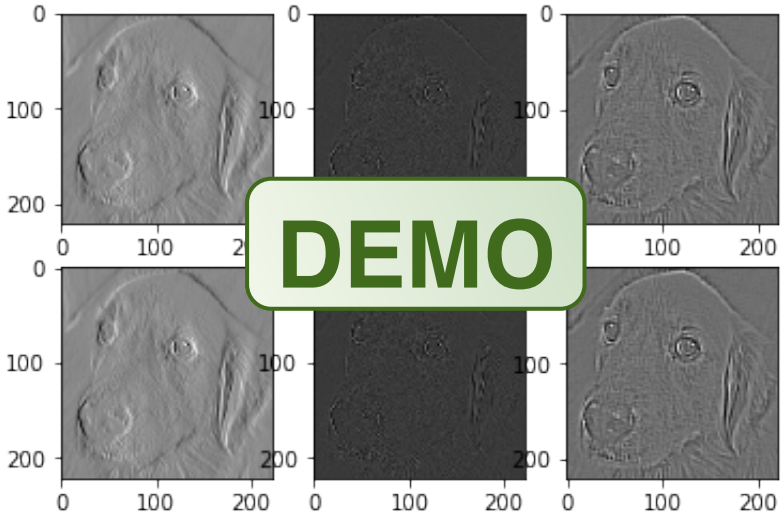
original image	filtered_image	filter size	padding	bias?
problems["img"][0]	problems["img_f"][0]	3x3	valid	no
problems["img"][1]	problems["img_f"][1]	3x3	valid	no
problems["img"][2]	problems["img_f"][2]	5x5	valid	no

Find out the three unknown filters. Try using different optimization methods, loss functions etc and see their effects on the result. Think about why different configurations work differently.

Hints and Notes

1. There's no need to do this assignment on GPU machines.
2. Learning rate and number of epochs have significant effect on the results, try tuning these parameters.
3. You can validate your filters by using a `validate()` function. If the output two rows look similar, then your filter is probably OK.

```
from assign2_utils import validate
validate('Axxxxxxxx.pkl')
```



Submit Your Results

Your answer should be a Python dictionary in the following format and you should dump it to a file using `pickle`. Please note that **you will get 0 mark if your file cannot be correctly loaded by the auto-grading program**.

```
answer = {
    # your name as shown in IVLE
    'Name': 'XU ZIWEI',
    # your Matriculation Number, starting with letter 'A'
    'MatricNum': 'AXXXXXXXX',
    # do check the size of filters
    'answer': {'filter': [ first_filter, second_filter, third_filter ] }
 }
import pickle as pk
with open('Axxxxxxxx.pkl', 'wb') as f:
    pk.dump(answer, f)
```

Include your validation image in the report.

For other submission instructions, please refer to the document of part 2.