Multiple Input and Output Channels

Objective for this Notebook

1. Learn on Multiple Input and Multiple Output Channels.

Table of Contents

In this lab, you will study convolution and review how the different operations change the relationship between input and output.

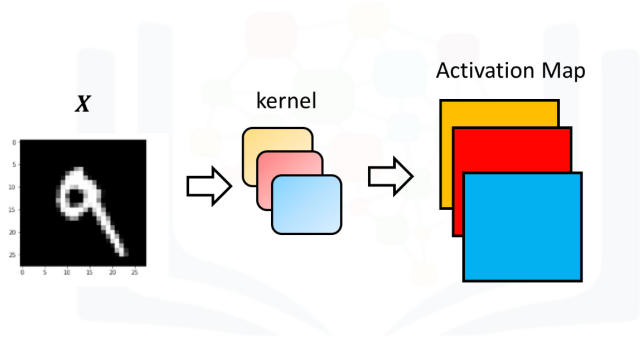
Estimated Time Needed: **25 min**

Import the following libraries:

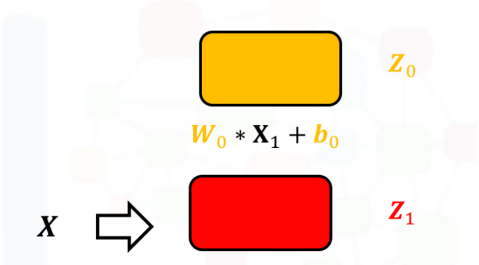
```
[ ]: import torch
import torch.nn as nn
import matplotlib.pyplot as plt
import numpy as np
from scipy import ndimage, misc
```

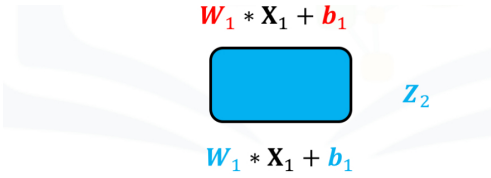
Multiple Output Channels

In Pytroch, you can create a `Conv2d` object with multiple outputs. For each channel, a kernel is created, and each kernel performs a convolution independently. As a result, the number of outputs is equal to the number of channels. This is demonstrated in the following figure. The number 9 is convolved with three kernels: each of a different color. There are three different activation maps represented by the different colors.



Symbolically, this can be represented as follows:





Create a `Conv2d` with three channels:

```
[ ]: conv1 = nn.Conv2d(in_channels=1, out_channels=3,kernel_size=3)
```

Pytorch randomly assigns values to each kernel. However, use kernels that have been developed to detect edges:

```
[ ]: Gx=torch.tensor([[1.0,0,-1.0],[2.0,0,-2.0],[1.0,0.0,-1.0]])
     Gy=torch.tensor([[1.0,2.0,1.0],[0.0,0.0,0.0],[-1.0,-2.0,-1.0]])

     conv1.state_dict()['weight'][0][0]=Gx
     conv1.state_dict()['weight'][1][0]=Gy
     conv1.state_dict()['weight'][2][0]=torch.ones(3,3)
```

Each kernel has its own bias, so set them all to zero:

```
[ ]: conv1.state_dict()['bias'][:]=torch.tensor([0.0,0.0,0.0])
     conv1.state_dict()['bias']
```

Print out each kernel:

```
[ ]: for x in conv1.state_dict()['weight']:
         print(x)
```

Create an input `image` to represent the input X:

```
[ ]: image=torch.zeros(1,1,5,5)
     image[0,0,:,2]=1
     image
```

Plot it as an image:

```
[ ]: plt.imshow(image[0,0,:,:].numpy(), interpolation='nearest', cmap=plt.cm.gray)
     plt.colorbar()
     plt.show()
```

Perform convolution using each channel:

```
[ ]: out=conv1(image)
```

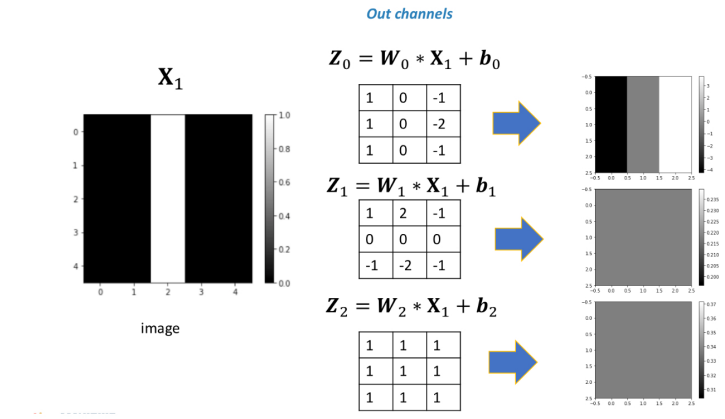
The result is a 1x3x3x3 tensor. This represents one sample with three channels, and each channel contains a 3x3 image. The same rules that govern the shape of each image were discussed in the last section.

```
[ ]: out.shape
```

Print out each channel as a tensor or an image:

```
[ ]: for channel,image in enumerate(out[0]):
         plt.imshow(image.detach().numpy(), interpolation='nearest', cmap=plt.cm.gray)
         print(image)
         plt.title("channel {}".format(channel))
         plt.colorbar()
         plt.show()
```

Different kernels can be used to detect various features in an image. You can see that the first channel fluctuates, and the second two channels produce a constant value. The following figure summarizes the process:



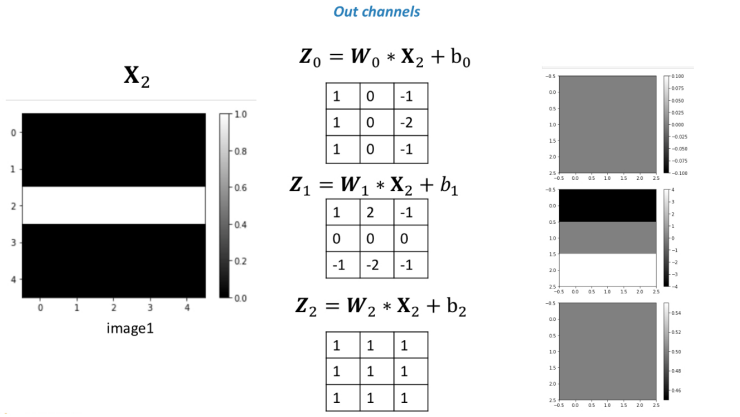
If you use a different image, the result will be different:

```
[ ]: image1=torch.zeros(1,1,5,5)
     image1[0,0,2,:]=1
     print(image1)
     plt.imshow(image1[0,0,:,:].detach().numpy(), interpolation='nearest', cmap=plt.cm.gray)
     plt.show()
```

In this case, the second channel fluctuates, and the first and the third channels produce a constant value.

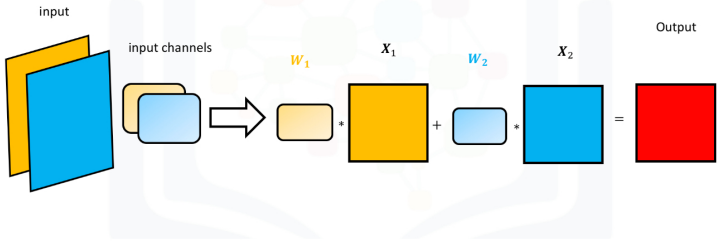
```
[ ]: out1=conv1(image1)
     for channel,image in enumerate(out1[0]):
         plt.imshow(image.detach().numpy(), interpolation='nearest', cmap=plt.cm.gray)
         print(image)
         plt.title("channel {}".format(channel))
         plt.colorbar()
         plt.show()
```

The following figure summarizes the process:



Multiple Input Channels

For two inputs, you can create two kernels. Each kernel performs a convolution on its associated input channel. The resulting output is added together as shown:



Create an input with two channels:

```
image2=torch.zeros(1,2,5,5)
image2[0,0,2,:]=-2
image2[0,1,2,:]=1
image2
```

Plot out each image:

```
for channel,image in enumerate(image2[0]):
    plt.imshow(image.detach().numpy(), interpolation='nearest', cmap=plt.cm.gray)
    print(image)
    plt.title("channel {}".format(channel))
    plt.colorbar()
    plt.show()
```

Create a `Conv2d` object with two inputs:

```
conv3 = nn.Conv2d(in_channels=2, out_channels=1,kernel_size=3)
```

Assign kernel values to make the math a little easier:

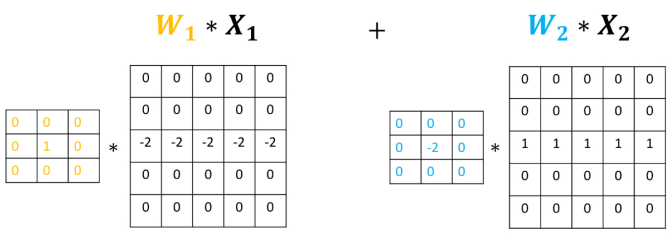
```
Gx1=torch.tensor([[0.0,0.0,0.0,0.0,0.0],[0,1.0,0],[0.0,0.0,0.0,0.0]])
conv3.state_dict()['weight'][0][0]=1*Gx1
conv3.state_dict()['weight'][0][1]=-2*Gx1
conv3.state_dict()['bias'][:]=torch.tensor([0.0])
```

```
conv3.state_dict()['weight']
```

Perform the convolution:

```
conv3(image2)
```

The following images summarize the process. The object performs Convolution.



Then, it adds the result:

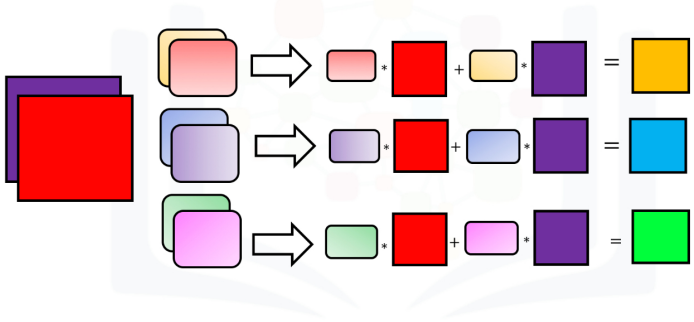
$$\mathbf{Z} = W_1 * X_1 + W_2 * X_2$$

$$\begin{bmatrix}0&0&0\\-2&-2&-2\\0&0&0\end{bmatrix}+\begin{bmatrix}0&0&0\\-2&-2&-2\\0&0&0\end{bmatrix}=\begin{bmatrix}0+0&0+0&0+0\\-2-2&-2-2&-2-2\\0+0&0+0&0+0\end{bmatrix}=\begin{bmatrix}0&0&0\\-4&-4&-4\\0&0&0\end{bmatrix}$$

Multiple Input and Multiple Output Channels

When using multiple inputs and outputs, a kernel is created for each input, and the process is repeated for each output. The process is summarized in the following image.

There are two input channels and 3 output channels. For each channel, the input in red and purple is convolved with an individual kernel that is colored differently. As a result, there are three outputs.



Create an example with two inputs and three outputs and assign the kernel values to make the math a little easier:

```
[ ]: conv4 = nn.Conv2d(in_channels=2, out_channels=3,kernel_size=3)
conv4.state_dict()['weight'][0][0]=torch.tensor([[0.0,0.0,0.0],[0,0.5,0],[0.0,0.0,0.0]])
conv4.state_dict()['weight'][0][1]=torch.tensor([[0.0,0.0,0.0],[0,0.5,0],[0.0,0.0,0.0]])


conv4.state_dict()['weight'][1][0]=torch.tensor([[0.0,0.0,0.0],[0,1,0],[0.0,0.0,0.0]])
conv4.state_dict()['weight'][1][1]=torch.tensor([[0.0,0.0,0.0],[0,-1,0],[0.0,0.0,0.0]])

conv4.state_dict()['weight'][2][0]=torch.tensor([[1.0,0,-1.0],[2.0,0,-2.0],[1.0,0.0,-1.0]])
conv4.state_dict()['weight'][2][1]=torch.tensor([[1.0,2.0,1.0],[0.0,0.0,0.0],[-1.0,-2.0,-1.0]])
```

For each output, there is a bias, so set them all to zero:

```
[ ]: conv4.state_dict()['bias'][:]=torch.tensor([0.0,0.0,0.0])
```

Create a two-channel image and plot the results:

```
[ ]: image4=torch.zeros(1,2,5,5)
image4[0][0]=torch.ones(5,5)
image4[0][1][2][2]=1
for channel,image in enumerate(image4[0]):
    plt.imshow(image.detach().numpy(), interpolation='nearest', cmap=plt.cm.gray)
    print(image)
    plt.title("channel {}".format(channel))
    plt.colorbar()
    plt.show()
```

Perform the convolution:

```
[ ]: z=conv4(image4)
z
```

The output of the first channel is given by:

$$Z_0 = W_{0,0} * X_0 + W_{0,1} * X_1$$

$$W_{0,0}=\begin{bmatrix}0&0&0\\0&0.5&0\\0&0&0\end{bmatrix},\quad X_0=\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix}$$

$$W_{0,1}=\begin{bmatrix}0&0&0\\0&0.5&0\\0&0&0\end{bmatrix},\quad X_1=\begin{bmatrix}0&0&0&0&0\\0&0&0&0&0\\0&0&1&0&0\\0&0&0&0&0\\0&0&0&0&0\end{bmatrix}$$

$$=\begin{bmatrix}0.5&0.5&0.5\\0.5&0.5&0.5\\0.5&0.5&0.5\end{bmatrix}+\begin{bmatrix}0&0&0\\0&0.5&0\\0&0&0\end{bmatrix}=\begin{bmatrix}0.5&0.5&0.5\\0.5&1&0.5\\0.5&0.5&0.5\end{bmatrix}$$

The output of the second channel is given by:

$$Z_1 = W_{1,0} * X_0 + W_{1,1} * X_1$$

$$W_{1,0}=\begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix},\quad X_0=\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix}$$

$$W_{1,1}=\begin{bmatrix}0&0&0\\0&-1&0\\0&0&0\end{bmatrix},\quad X_1=\begin{bmatrix}0&0&0&0&0\\0&0&0&0&0\\0&0&1&0&0\\0&0&0&0&0\\0&0&0&0&0\end{bmatrix}$$

1	1	1	1	1

0	0	0	0	0

$$= \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

The output of the third channel is given by:

$$Z_2 = W_{2,0} * X_0 + W_{2,1} * X_1$$

$$W_{2,0} = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad X_0 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$W_{2,1} = \begin{bmatrix} 1 & 2 & -1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad X_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Practice Questions

Use the following two convolution objects to produce the same result as two input channel convolution on imageA and imageB as shown in the following image:

[ ]:
```
imageA=torch.zeros(1,1,5,5)
imageB=torch.zeros(1,1,5,5)
imageA[0,0,2,:]=-2
imageB[0,0,2,:]=1


conv5 = nn.Conv2d(in_channels=1, out_channels=1,kernel_size=3)
conv6 = nn.Conv2d(in_channels=1, out_channels=1,kernel_size=3)


Gx1=torch.tensor([[0.0,0.0,0.0,0.0,0.0],[0,1.0,0],[0.0,0.0,0.0,0.0]])
conv5.state_dict()['weight'][0][0]=1*Gx1
conv6.state_dict()['weight'][0][0]=-2*Gx1
conv5.state_dict()['bias'][:]=torch.tensor([0.0])
conv6.state_dict()['bias'][:]=torch.tensor([0.0])
```

$$W_1 * X_1 + W_2 * X_2$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & -2 & -2 & -2 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

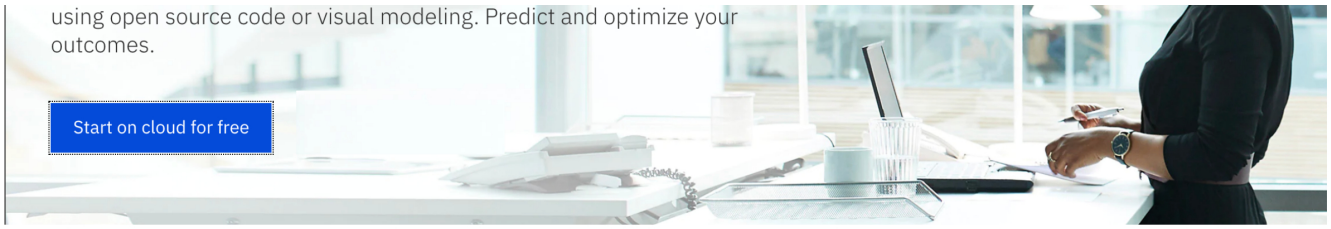
$$Z = W_1 * X_1 + W_2 * X_2$$

$$\begin{bmatrix} 0 & 0 & 0 \\ -2 & -2 & -2 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ -2 & -2 & -2 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0+0 & 0+0 & 0+0 \\ -2-2 & -2-2 & -2-2 \\ 0+0 & 0+0 & 0+0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 \\ -4 & -4 & -4 \\ 0 & 0 & 0 \end{bmatrix}$$

Double-click **here** for the solution.

<!-- Your answer is below:
conv5(imageA)+conv6(imageB)
-->



About the Authors:

Joseph Santarcangelo has a PhD in Electrical Engineering. His research focused on using machine learning, signal processing, and computer vision to determine how videos impact human cognition.

Other contributors: Michelle Carey, Mavis Zhou

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2020-09-23	2.0	Srishti	Migrated Lab to Markdown and added to course repo in GitLab