9 3Multiple Channel Convolution

November 1, 2025

Multiple Input and Output Channels

Objective for this Notebook

1. Learn on Multiple Input and Multiple Output Channels.

1 Table of Contents

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

Multiple Output Channels

Multiple Input Channels

Multiple Input and Multiple Output Channels

Practice Questions

Estimated Time Needed: 25 min

Import the following libraries:

[1]:
```
%%time
%pip install pandas numpy matplotlib scipy
%pip install torch==2.8.0+cpu torchvision==0.23.0+cpu torchaudio==2.8.0+cpu \
    --index-url https://download.pytorch.org/whl/cpu
```

```
Collecting pandas
  Downloading pandas-2.3.3-cp312-cp312-
manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl.metadata (91 kB)
Collecting numpy
  Downloading
numpy-2.3.4-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl.metadata
(62 kB)
Collecting matplotlib
  Downloading matplotlib-3.10.7-cp312-cp312-
manylinux2014_x86_64.manylinux_2_17_x86_64.whl.metadata (11 kB)
Collecting scipy
  Downloading
scipy-1.16.3-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.whl.metadata
(62 kB)
```

```
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/conda/lib/python3.12/site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.12/site-
packages (from pandas) (2024.2)
Collecting tzdata>=2022.7 (from pandas)
  Downloading tzdata-2025.2-py2.py3-none-any.whl.metadata (1.4 kB)
Collecting contourpy>=1.0.1 (from matplotlib)
  Downloading contourpy-1.3.3-cp312-cp312-
manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl.metadata (5.5 kB)
Collecting cycler>=0.10 (from matplotlib)
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib)
  Downloading fonttools-4.60.1-cp312-cp312-
manylinux1_x86_64.manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_5_x86_6
4.whl.metadata (112 kB)
Collecting kiwisolver>=1.3.1 (from matplotlib)
  Downloading kiwisolver-1.4.9-cp312-cp312-
manylinux2014_x86_64.manylinux_2_17_x86_64.whl.metadata (6.3 kB)
Requirement already satisfied: packaging>=20.0 in
/opt/conda/lib/python3.12/site-packages (from matplotlib) (24.2)
Collecting pillow>=8 (from matplotlib)
  Downloading pillow-12.0.0-cp312-cp312-
manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl.metadata (8.8 kB)
Collecting pyparsing>=3 (from matplotlib)
  Downloading pyparsing-3.2.5-py3-none-any.whl.metadata (5.0 kB)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-
packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Downloading
pandas-2.3.3-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl (12.4
MB)
                         12.4/12.4 MB
158.0 MB/s eta 0:00:00
Downloading
numpy-2.3.4-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl (16.6
MB)
                         16.6/16.6 MB
182.8 MB/s eta 0:00:00
Downloading
matplotlib-3.10.7-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.whl
(8.7 MB)
                         8.7/8.7 MB
102.0 MB/s eta 0:00:00
Downloading
scipy-1.16.3-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.whl (35.7
MB)
                         35.7/35.7 MB
117.9 MB/s eta 0:00:00a 0:00:01
Downloading
```

```
contourpy-1.3.3-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl (362
kB)
Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
Downloading fonttools-4.60.1-cp312-cp312-
manylinux1_x86_64.manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_5_x86_6
4.whl (4.9 MB)
                         4.9/4.9 MB
85.1 MB/s eta 0:00:00
Downloading
kiwisolver-1.4.9-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.whl (1.5
MB)
                         1.5/1.5 MB
49.2 MB/s eta 0:00:00
Downloading
pillow-12.0.0-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl (7.0
MB)
                         7.0/7.0 MB
49.8 MB/s eta 0:00:00
Downloading pyparsing-3.2.5-py3-none-any.whl (113 kB)
Downloading tzdata-2025.2-py2.py3-none-any.whl (347 kB)
Installing collected packages: tzdata, pyparsing, pillow, numpy, kiwisolver,
fonttools, cycler, scipy, pandas, contourpy, matplotlib
Successfully installed contourpy-1.3.3 cycler-0.12.1 fonttools-4.60.1
kiwisolver-1.4.9 matplotlib-3.10.7 numpy-2.3.4 pandas-2.3.3 pillow-12.0.0
pyparsing-3.2.5 scipy-1.16.3 tzdata-2025.2
Note: you may need to restart the kernel to use updated packages.
Looking in indexes: https://download.pytorch.org/whl/cpu
Collecting torch==2.8.0+cpu
  Downloading https://download.pytorch.org/whl/cpu/torch-2.8.0%2Bcpu-
cp312-cp312-manylinux_2_28_x86_64.whl.metadata (29 kB)
Collecting torchvision==0.23.0+cpu
  Downloading https://download.pytorch.org/whl/cpu/torchvision-0.23.0%2Bcpu-
cp312-cp312-manylinux_2_28_x86_64.whl.metadata (6.1 kB)
Collecting torchaudio==2.8.0+cpu
  Downloading https://download.pytorch.org/whl/cpu/torchaudio-2.8.0%2Bcpu-
cp312-cp312-manylinux_2_28_x86_64.whl.metadata (7.2 kB)
Collecting filelock (from torch==2.8.0+cpu)
  Downloading https://download.pytorch.org/whl/filelock-3.19.1-py3-none-
any.whl.metadata (2.1 kB)
Requirement already satisfied: typing-extensions>=4.10.0 in
/opt/conda/lib/python3.12/site-packages (from torch==2.8.0+cpu) (4.12.2)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.12/site-
packages (from torch==2.8.0+cpu) (75.8.0)
Collecting sympy>=1.13.3 (from torch==2.8.0+cpu)
  Downloading https://download.pytorch.org/whl/sympy-1.14.0-py3-none-
any.whl.metadata (12 kB)
Collecting networkx (from torch==2.8.0+cpu)
  Downloading https://download.pytorch.org/whl/networkx-3.5-py3-none-
```

```
any.whl.metadata (6.3 kB)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.12/site-packages
(from torch==2.8.0+cpu) (3.1.5)
Collecting fsspec (from torch==2.8.0+cpu)
  Downloading https://download.pytorch.org/whl/fsspec-2025.9.0-py3-none-
any.whl.metadata (10 kB)
Requirement already satisfied: numpy in /opt/conda/lib/python3.12/site-packages
(from torchvision==0.23.0+cpu) (2.3.4)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in
/opt/conda/lib/python3.12/site-packages (from torchvision==0.23.0+cpu) (12.0.0)
Collecting mpmath<1.4,>=1.1.0 (from sympy>=1.13.3->torch==2.8.0+cpu)
  Downloading https://download.pytorch.org/whl/mpmath-1.3.0-py3-none-any.whl
(536 kB)
                           536.2/536.2 kB
772.1 kB/s eta 0:00:00m-:--:--
Requirement already satisfied: MarkupSafe>=2.0 in
/opt/conda/lib/python3.12/site-packages (from jinja2->torch==2.8.0+cpu) (3.0.2)
Downloading https://download.pytorch.org/whl/cpu/torch-2.8.0%2Bcpu-
cp312-cp312-manylinux_2_28_x86_64.whl (183.9 MB)
                           183.9/183.9 MB
6.9 MB/s eta 0:00:0000:0100:01
Downloading https://download.pytorch.org/whl/cpu/torchvision-0.23.0%2Bcpu-
cp312-cp312-manylinux_2_28_x86_64.whl (1.9 MB)
                           1.9/1.9 MB
2.7 MB/s eta 0:00:00a 0:00:01m
Downloading https://download.pytorch.org/whl/cpu/torchaudio-2.8.0%2Bcpu-
cp312-cp312-manylinux_2_28_x86_64.whl (1.8 MB)
                           1.8/1.8 MB
25.5 MB/s eta 0:00:00
Downloading https://download.pytorch.org/whl/sympy-1.14.0-py3-none-any.whl (6.3
MB)
                           6.3/6.3 MB
1.6 MB/s eta 0:00:0000:0100:01m
Downloading https://download.pytorch.org/whl/filelock-3.19.1-py3-none-any.whl
(15 kB)
Downloading https://download.pytorch.org/whl/fsspec-2025.9.0-py3-none-any.whl
(199 kB)
Downloading https://download.pytorch.org/whl/networkx-3.5-py3-none-any.whl (2.0
MB)
                           2.0/2.0 MB
3.2 MB/s eta 0:00:00-:--:--
Installing collected packages: mpmath, sympy, networkx, fsspec, filelock, torch,
torchvision, torchaudio
Successfully installed filelock-3.19.1 fsspec-2025.9.0 mpmath-1.3.0 networkx-3.5
sympy-1.14.0 torch-2.8.0+cpu torchaudio-2.8.0+cpu torchvision-0.23.0+cpu
Note: you may need to restart the kernel to use updated packages.
CPU times: user 1.55 s, sys: 384 ms, total: 1.93 s
Wall time: 1min 46s
```

[2]:
```
import torch
import torch.nn as nn
import matplotlib.pyplot as plt
import numpy as np
from scipy import ndimage, misc
```

```
/tmp/ipykernel_298/3053585304.py:5: DeprecationWarning: scipy.misc is deprecated
and will be removed in 2.0.0
  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:

[3]:
```
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:

[4]:
```
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:

[5]:
```
conv1.state_dict()['bias'][:]=torch.tensor([0.0,0.0,0.0])
conv1.state_dict()['bias']
```

[5]:
```
tensor([0., 0., 0.])
```

Print out each kernel:

[6]:
```
for x in conv1.state_dict()['weight']:
    print(x)
```

```
tensor([[[ 1.,  0., -1.],
         [ 2.,  0., -2.],
         [ 1.,  0., -1.]]])
tensor([[[ 1.,  2.,  1.],
         [ 0.,  0.,  0.],
         [-1., -2., -1.]]])
```

```
tensor([[[1., 1., 1.],
         [1., 1., 1.],
         [1., 1., 1.]]])
```

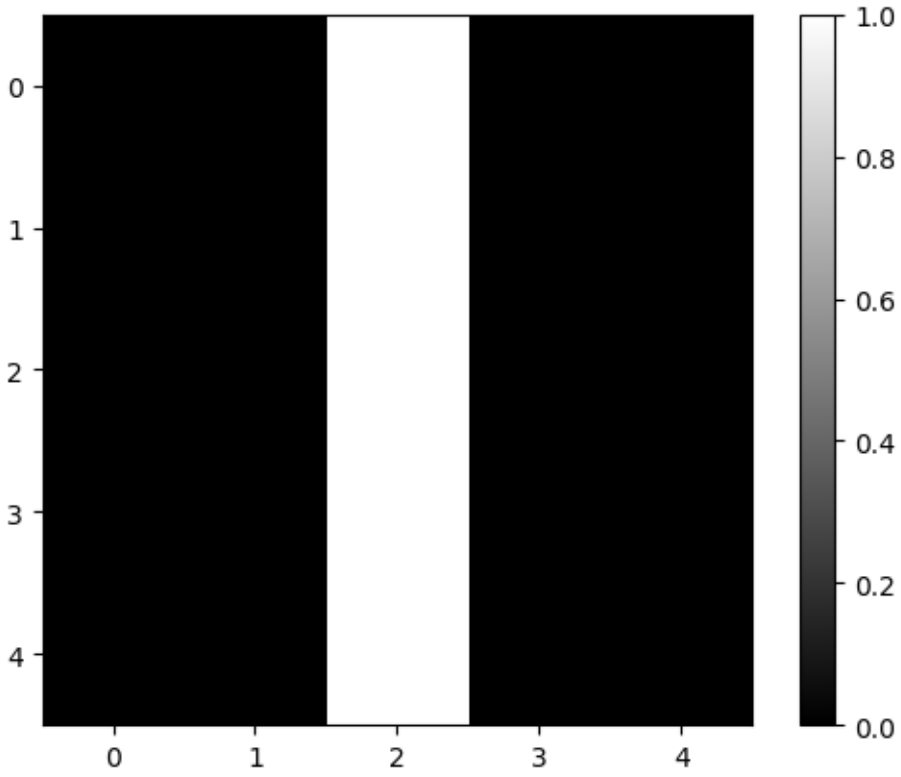
Create an input image to represent the input X:

[7]:
```
image=torch.zeros(1,1,5,5)
image[0,0,:,2]=1
image
```

[7]:
```
tensor([[[[0., 0., 1., 0., 0.],
          [0., 0., 1., 0., 0.],
          [0., 0., 1., 0., 0.],
          [0., 0., 1., 0., 0.],
          [0., 0., 1., 0., 0.]]]])
```

Plot it as an image:

[8]:
```
plt.imshow(image[0,0,:,:].numpy(), interpolation='nearest', cmap=plt.cm.gray)
plt.colorbar()
plt.show()
```



Perform convolution using each channel:

[9]:
```
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.

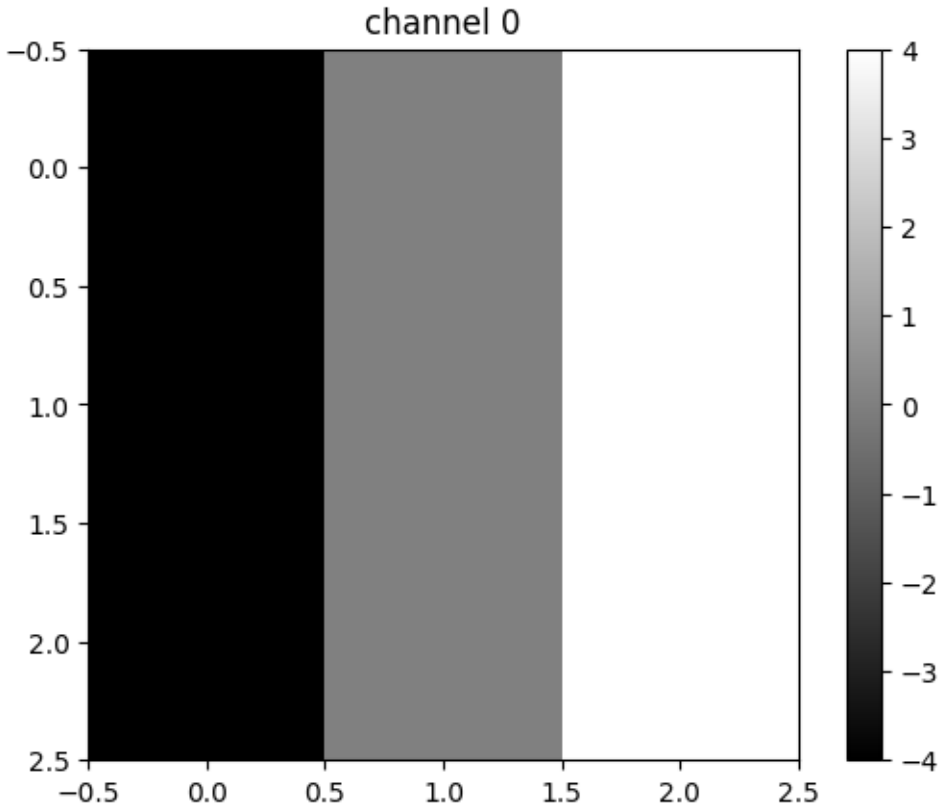
[10]:
```
out.shape
```

[10]:
```
torch.Size([1, 3, 3, 3])
```

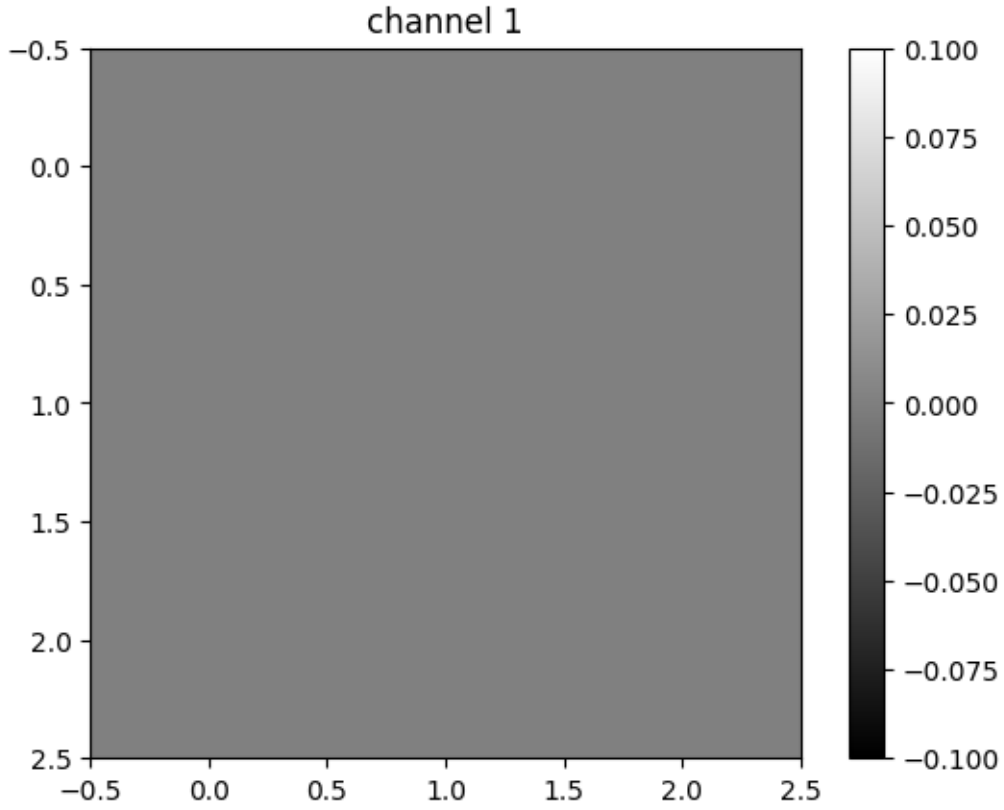
Print out each channel as a tensor or an image:

[11]:
```
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()
```

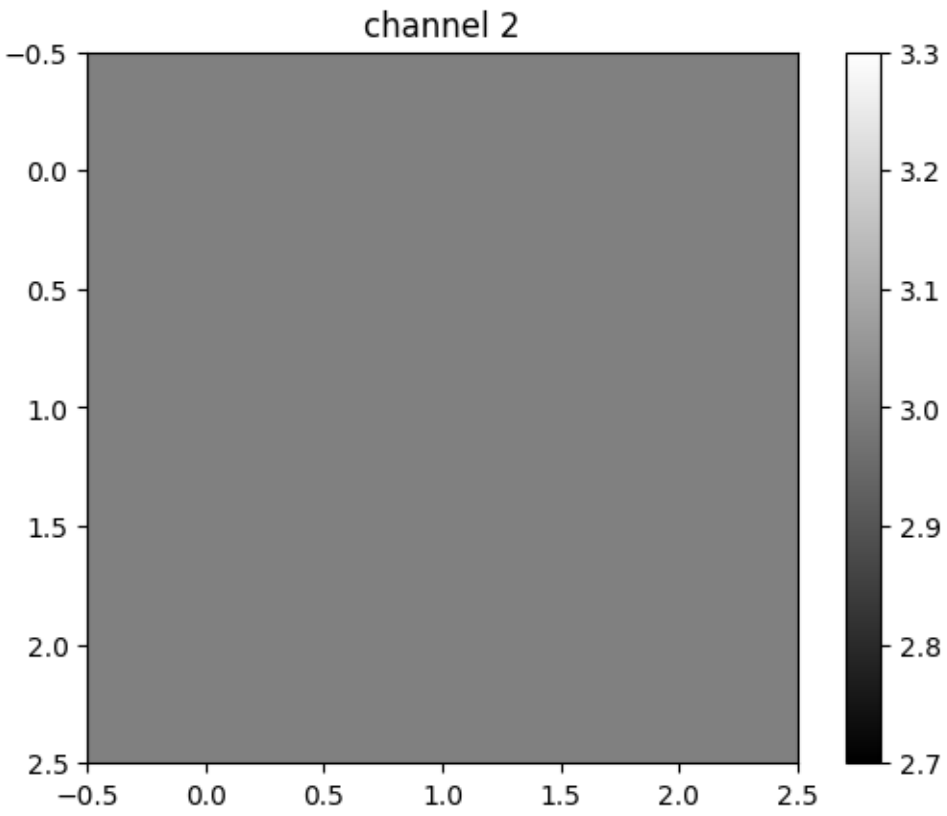
```
tensor([[-4.,  0.,  4.],
        [-4.,  0.,  4.],
        [-4.,  0.,  4.]], grad_fn=<UnbindBackward0>)
```



```
tensor([[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]], grad_fn=<UnbindBackward0>)
```




```
tensor([[3., 3., 3.],
        [3., 3., 3.],
        [3., 3., 3.]], grad_fn=<UnbindBackward0>)
```

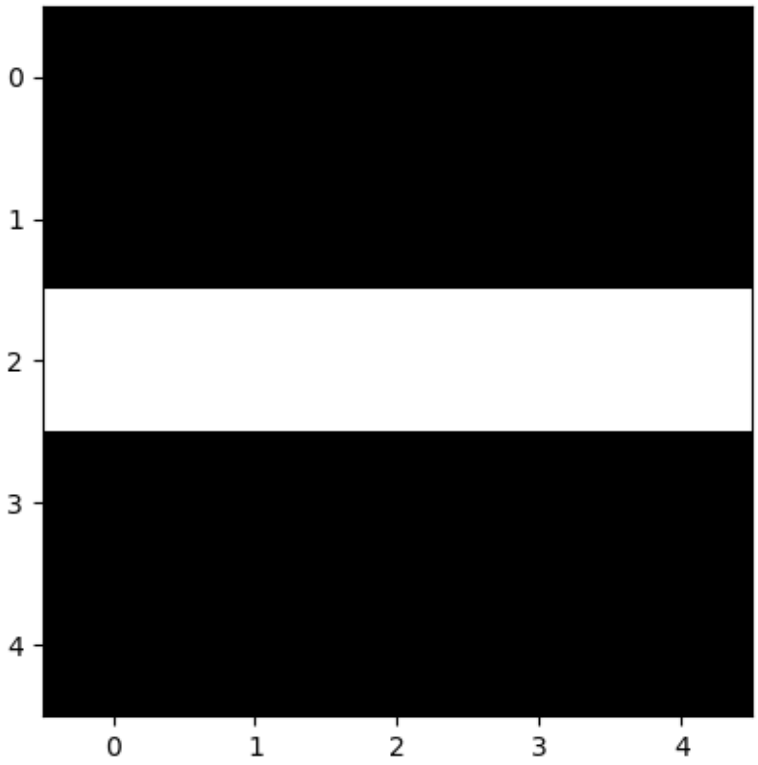


Different kernels can be used to detect various features in an image. You can see that the first channel fluctuates, and the second two channels produce a constant value. The following figure summarizes the process:

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

[12]:
```
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()
```

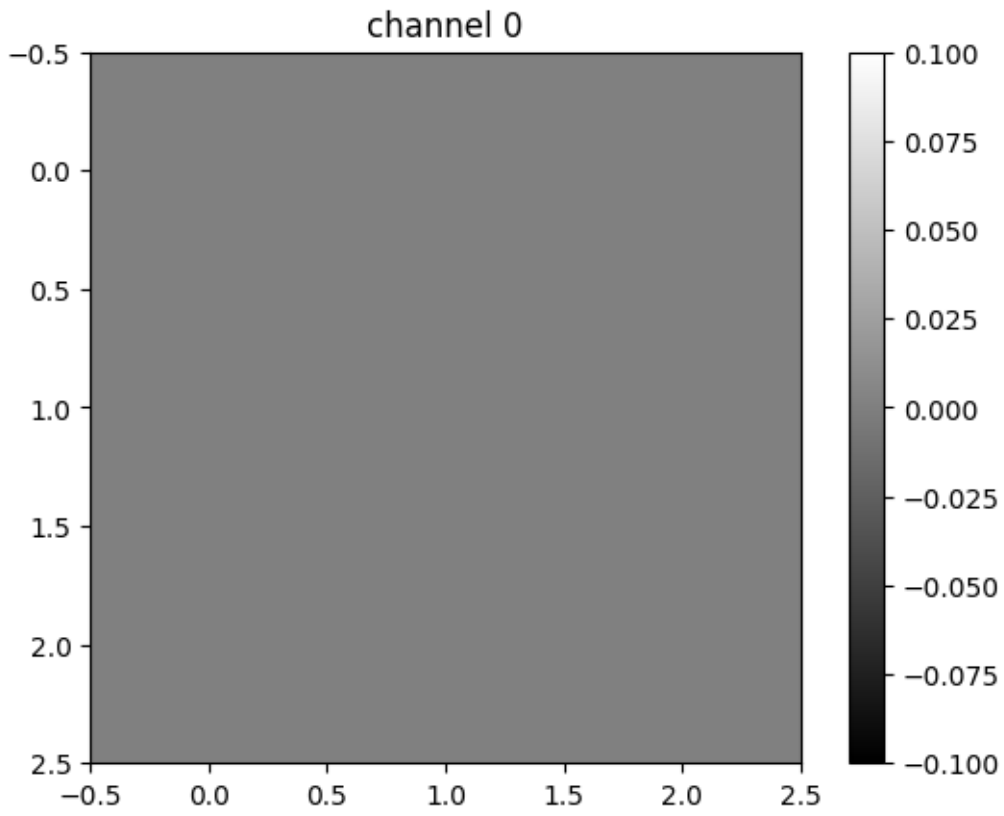
```
tensor([[[[0., 0., 0., 0., 0.],
          [0., 0., 0., 0., 0.],
          [1., 1., 1., 1., 1.],
          [0., 0., 0., 0., 0.],
          [0., 0., 0., 0., 0.]]]])
```



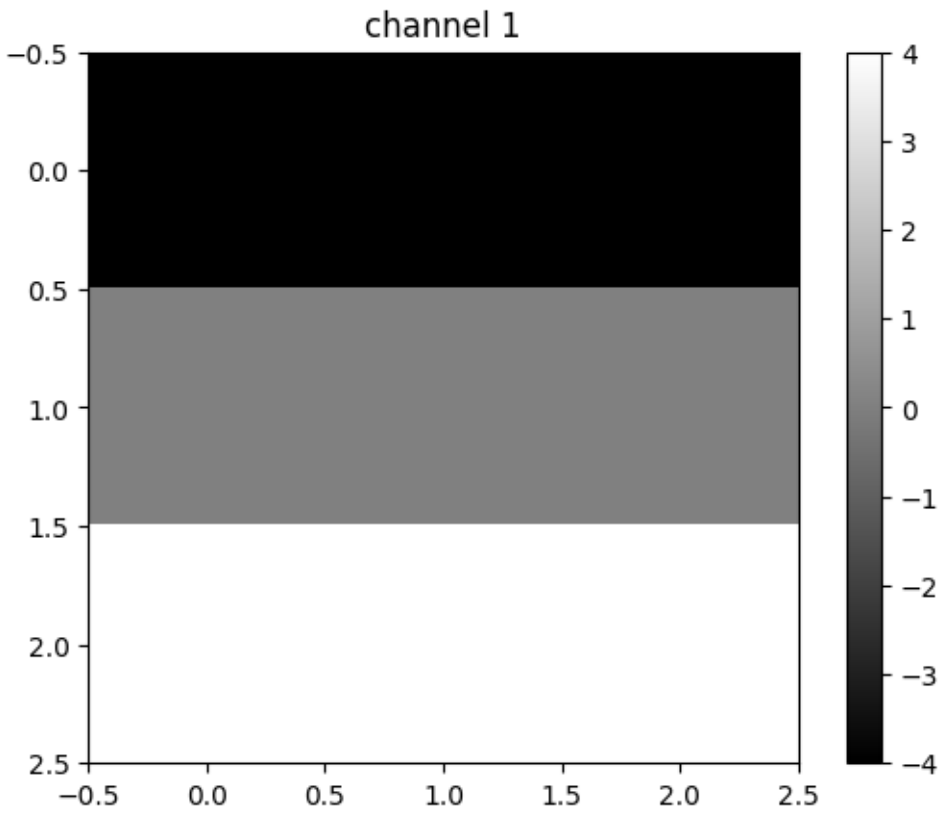
In this case, the second channel fluctuates, and the first and the third channels produce a constant value.

[13]:
```
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()
```

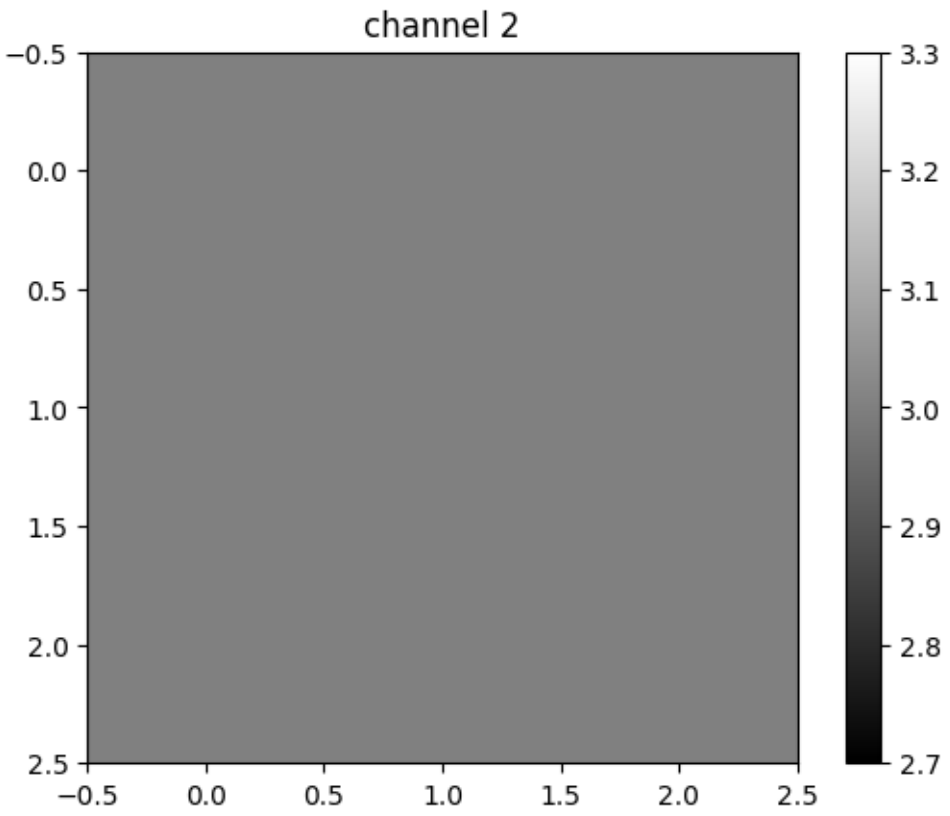
```
tensor([[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]], grad_fn=<UnbindBackward0>)
```




```
tensor([[-4., -4., -4.],
        [ 0.,  0.,  0.],
        [ 4.,  4.,  4.]], grad_fn=<UnbindBackward0>)
```




```
tensor([[3., 3., 3.],
        [3., 3., 3.],
        [3., 3., 3.]], grad_fn=<UnbindBackward0>)
```



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:

[14]:
```
image2=torch.zeros(1,2,5,5)
image2[0,0,2,:]=-2
image2[0,1,2,:]=1
image2
```

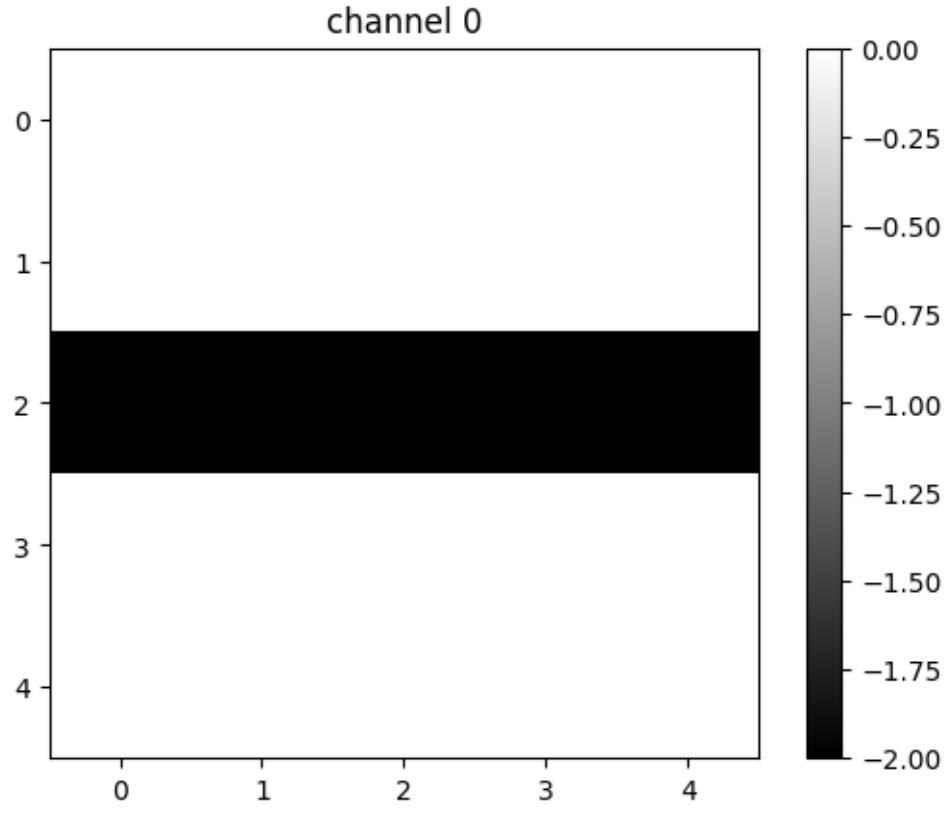
[14]:
```
tensor([[[[ 0.,  0.,  0.,  0.,  0.],
          [ 0.,  0.,  0.,  0.,  0.],
          [-2., -2., -2., -2., -2.],
          [ 0.,  0.,  0.,  0.,  0.],
          [ 0.,  0.,  0.,  0.,  0.]],

         [[ 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.]]]])
```

Plot out each image:

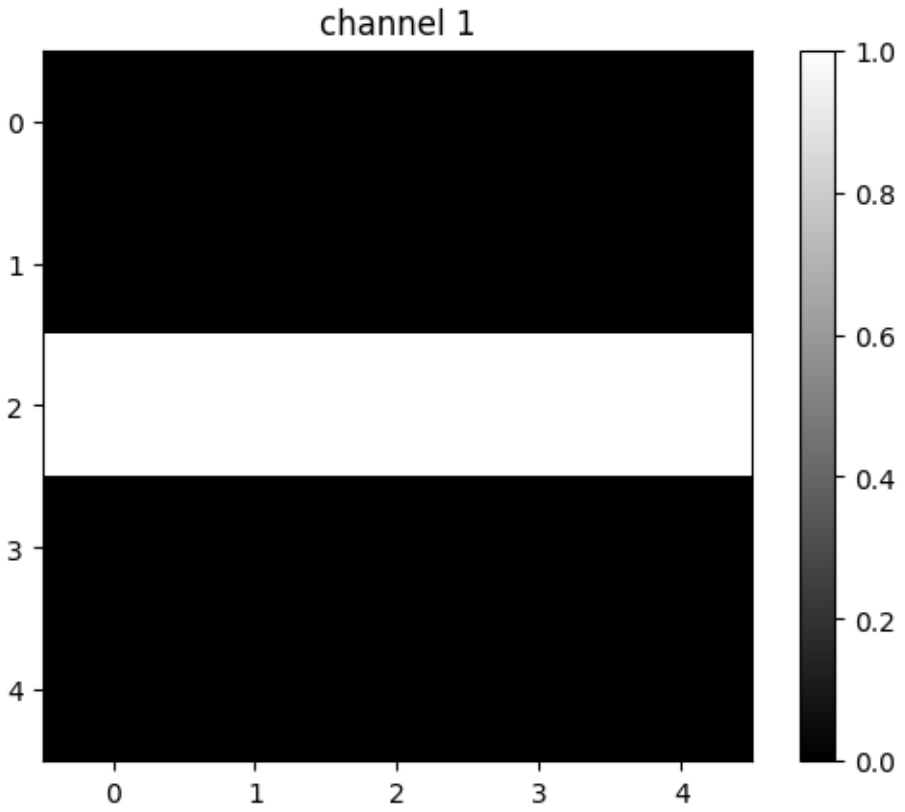
[15]:
```
for channel,image in enumerate(image2[0]):
    plt.imshow(image.detach().numpy(), interpolation='nearest', cmap=plt.cm.
 ↪gray)
    print(image)
    plt.title("channel {}".format(channel))
    plt.colorbar()
    plt.show()
```

```
tensor([[ 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.]])
```




```
tensor([[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.]])
```



Create a Conv2d object with two inputs:

[16]:
```
conv3 = nn.Conv2d(in_channels=2, out_channels=1,kernel_size=3)
```

Assign kernel values to make the math a little easier:

[17]:
```
Gx1=torch.tensor([[0.0,0.0,0.0],[0,1.0,0],[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])
```

[18]:
```
conv3.state_dict()['weight']
```

[18]:
```
tensor([[[[ 0.,  0.,  0.],
          [ 0.,  1.,  0.],
          [ 0.,  0.,  0.]],

         [[-0., -0., -0.],
          [-0., -2., -0.],
```

```
          [-0., -0., -0.]]]])
```

Perform the convolution:

[19]:
```
conv3(image2)
```

[19]:
```
tensor([[[[ 0.,  0.,  0.],
          [-4., -4., -4.],
          [ 0.,  0.,  0.]]]], grad_fn=<ConvolutionBackward0>)
```

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

Then, it adds the result:

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:

[20]:
```
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:

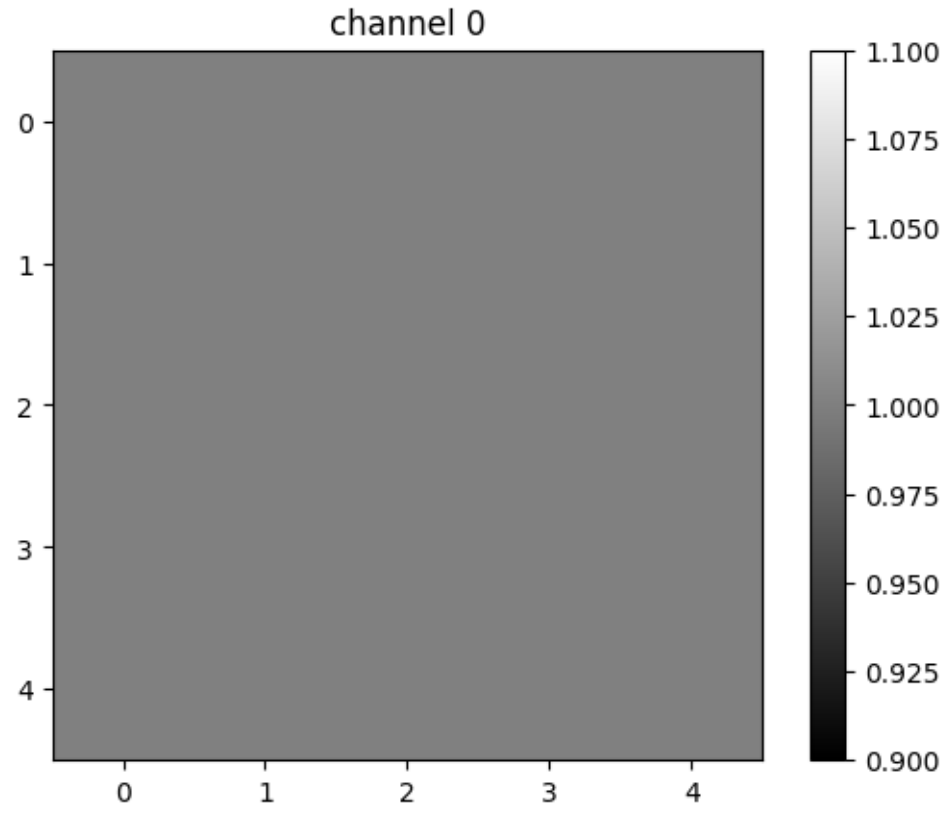
[21]:
```
conv4.state_dict()['bias'][:]=torch.tensor([0.0,0.0,0.0])
```

Create a two-channel image and plot the results:

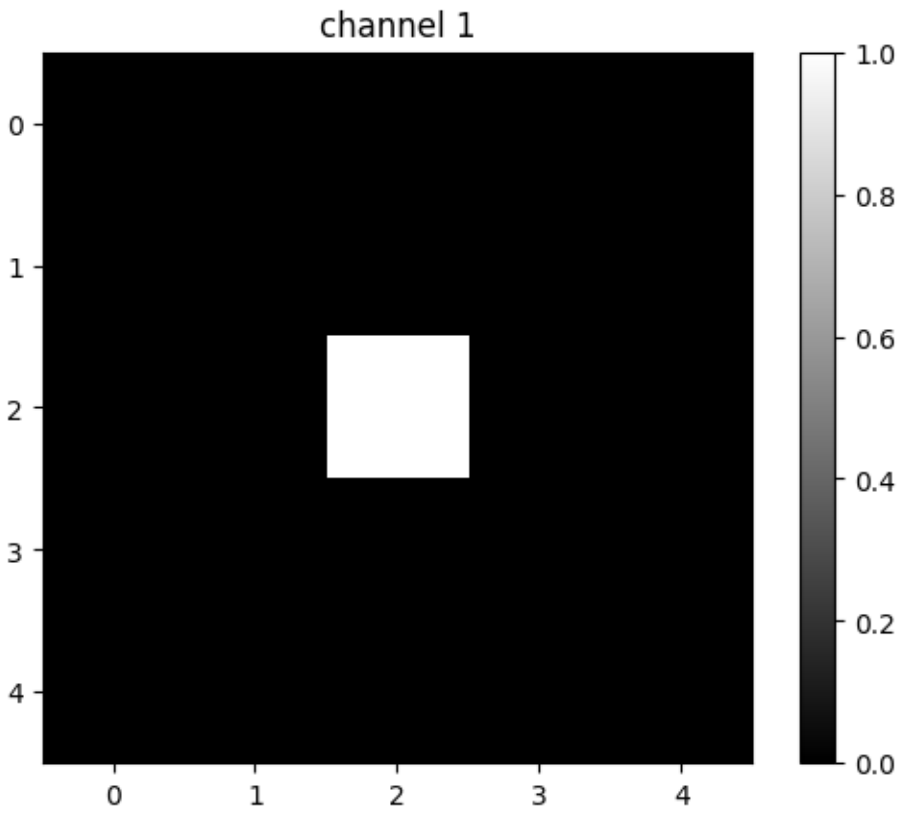
[22]:
```
image4=torch.zeros(1,2,5,5)
image4[0][0]=torch.ones(5,5)
```

```
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()
```

```
tensor([[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.]])
```




```
tensor([[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.]])
```



Perform the convolution:

[23]:
```
z=conv4(image4)
z
```

[23]:
```
tensor([[[[ 0.5000,  0.5000,  0.5000],
          [ 0.5000,  1.0000,  0.5000],
          [ 0.5000,  0.5000,  0.5000]],

         [[ 1.0000,  1.0000,  1.0000],
          [ 1.0000,  0.0000,  1.0000],
          [ 1.0000,  1.0000,  1.0000]],

         [[-1.0000, -2.0000, -1.0000],
          [ 0.0000,  0.0000,  0.0000],
          [ 1.0000,  2.0000,  1.0000]]]], grad_fn=<ConvolutionBackward0>)
```

The output of the first channel is given by:

The output of the second channel is given by:

The output of the third channel is given by:

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:

[27]:
```
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],[0,1.0,0],[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])
conv5(imageA) + conv6(imageB)
```

[27]:
```
tensor([[[[ 0.,  0.,  0.],
          [-4., -4., -4.],
          [ 0.,  0.,  0.]]]], grad_fn=<AddBackward0>)
```

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

1.0.1 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

##