

CONVOLUTION

Input

$$\begin{bmatrix} 1 & 2 & 0 & 1 & 3 \\ 4 & 1 & 0 & 2 & 1 \\ 2 & 3 & 1 & 0 & 1 \\ 1 & 2 & 0 & 1 & 3 \\ 3 & 1 & 2 & 3 & 0 \end{bmatrix}$$

Kernal

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

Output

$$\begin{bmatrix} 6 & 3 & 2 \\ 5 & 2 & 1 \\ 4 & 1 & 0 \end{bmatrix}$$

$$\begin{aligned} \text{Output}(1,2) &= (2 \times 1) + (0 \times 0) + (1 \times -1) + (1 \times 1) + (0 \times 0) + (2 \times -1) + (3 \times 1) + (1 \times 0) + (0 \times -1) \\ &= 2 + 0 - 1 + 1 + 0 - 2 + 3 + 0 - 0 \\ &= 3 \end{aligned}$$

Pytorch.ipynb - Colab

Animations of Convolution and

Image Kernels explained visually

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=IC8-fpW_W2--

☆

Paused

+ Code + Text All changes saved

✓ RAM Disk

Convolution Process

```
[ ] from google.colab import drive

# Mount Google Drive
drive.mount('/content/drive')

Mounted at /content/drive
```

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image


# Function to perform convolution
def convolve2d(image, kernel):
    kernel_height, kernel_width = kernel.shape
    image_height, image_width = image.shape

    output_height = image_height - kernel_height + 1
    output_width = image_width - kernel_width + 1

    output = np.zeros((output_height, output_width))

    for i in range(output_height):
```

Connected to Python 3 Google Compute Engine backend

 AISCENCES

WWW.AISCENCES.IO

Odemy


 RAM 





Mounted at /content/drive



✓ Connected to Python 3 Google Compute Engine backend

Pytorch.ipynb - Colab

Animations of Convolution and

Image Kernels explained visually

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=PB1e_6aXW3Cf

RAM

Disk

Paused

+ Code

+ Text

All changes saved

[]

drive.mount('/content/drive')

Mounted at /content/drive

import numpy as np

import matplotlib.pyplot as plt

from PIL import Image

Function to perform convolution

def convolve2d(image, kernel):

kernel_height, kernel_width = kernel.shape

image_height, image_width = image.shape

output_height = image_height - kernel_height + 1

output_width = image_width - kernel_width + 1

output = np.zeros((output_height, output_width))

for i in range(output_height):

for j in range(output_width):

patch = image[i:i+kernel_height, j:j+kernel_width]

output[i, j] = np.sum(patch * kernel)

return output

[] # image_path = r"/content/drive/MyDrive/Dataset/images.jpeg"

Connected to Python 3 Google Compute Engine backend

AISCIENCES

WWW.AISCIENCES.IO

Odemy

Pytorch.ipynb - Colab

Animations of Convolution and

Image Kernels explained visually

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=PB1e_6aXW3Cf

RAM

Disk

Paused

+ Code

+ Text

All changes saved

Function to perform convolution

def convolve2d(image, kernel):

kernel_height, kernel_width = kernel.shape

image_height, image_width = image.shape

output_height = image_height - kernel_height + 1

output_width = image_width - kernel_width + 1

output = np.zeros((output_height, output_width))

for i in range(output_height):

for j in range(output_width):

patch = image[i:i+kernel_height, j:j+kernel_width]

output[i, j] = np.sum(patch * kernel)

return output

[] # image_path = r"/content/drive/MyDrive/Dataset/images.jpeg"

image = Image.open(image_path).convert('L') # Convert to grayscale

image = np.array(image)

image = np.random.rand(5, 5)

Define a kernel (e.g., edge detection)

kernel = np.array([

[1, 0, -1]

Connected to Python 3 Google Compute Engine backend

WWW.AISCIENCES.IO

AI SCIENCES

Odemy

Pytorch.ipynb - Colab

Animations of Convolution and

Image Kernels explained visually

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=xe7qidi5W289

☆

Paused

+ Code + Text All changes saved

✓ RAM Disk

[]

Apply convolution
convolved_image = convolve2d(image, kernel)

Plot the results
fig, axes = plt.subplots(1, 2, figsize=(15, 5))
axes[0].imshow(image, cmap='gray')
axes[0].set_title('Original Image')
axes[1].imshow(convolved_image, cmap='gray')
axes[1].set_title('Convolved Image')

↑ ↓ ↺ ⌨ ⚙ 📄 🗑 ⋮

0s

▶

image = np.random.rand(5, 5)
image

↩

array([[0.09865053, 0.52521243, 0.36051894, 0.21917541, 0.96039324],
 [0.36312466, 0.20557749, 0.09425741, 0.21215107, 0.04927506],
 [0.7737655 , 0.10031832, 0.44264185, 0.97337431, 0.33021274],
 [0.11717513, 0.63664965, 0.31029961, 0.4635026 , 0.76889252],
 [0.45942061, 0.89752827, 0.7664678 , 0.72507998, 0.83271031]])

[]

[]

[]

0s completed at 8:41 PM

AISCIENCES

WWW.AISCIENCES.IO

```
[2] image = np.random.rand(5, 5)
image
```

Pytorch.ipynb - Colab

Animations of Convolution and

Image Kernels explained visual

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=IC8-fpW_W2--

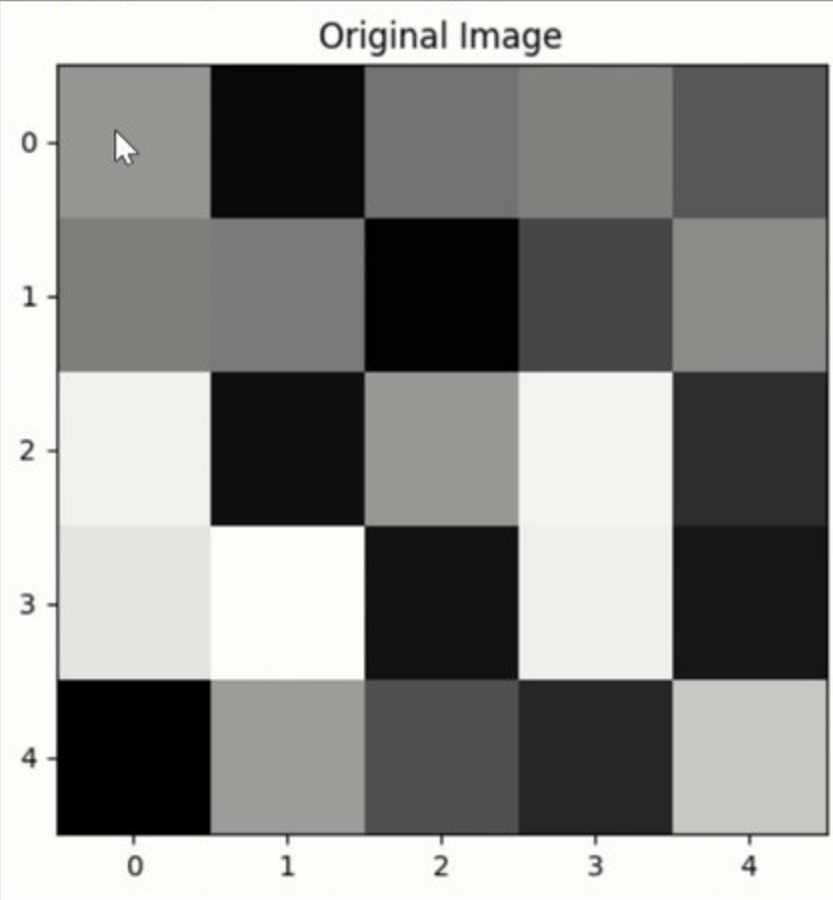
+ Code + Text

1s

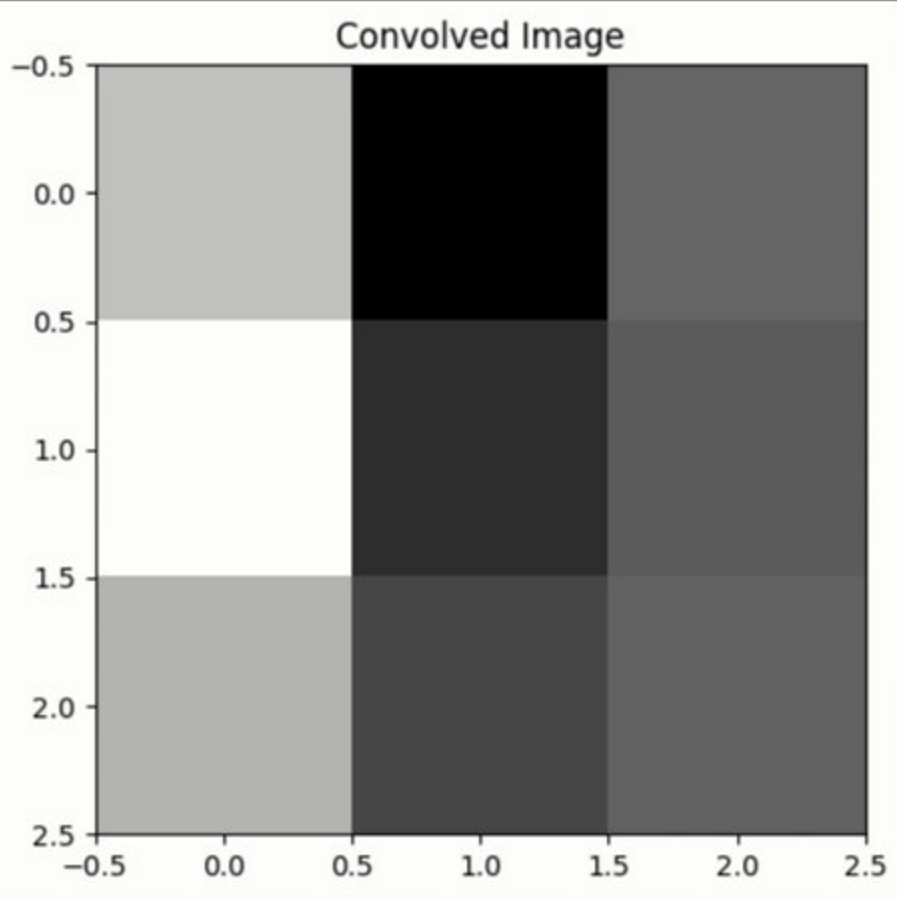
```
axes[1].imshow(convolved_image, cmap=gray)
axes[1].set_title('Convolved Image')

Text(0.5, 1.0, 'Convolved Image')
```

Original Image



Convolved Image



0s completed at 8:42 PM

WWW.AISCIENCES.IO

Odemy

Pytorch.ipynb - Colab

Animations of Convolution and

Image Kernels explained visually

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=mbNvxTAdmFg3

RAMDisk

Convolution Process

from google.colab import drive

Mount Google Drive

drive.mount('/content/drive')

Mounted at /content/drive

[1]

import numpy as np

import matplotlib.pyplot as plt

from PIL import Image

Function to perform convolution

def convolve2d(image, kernel):

kernel_height, kernel_width = kernel.shape

image_height, image_width = image.shape

output_height = image_height - kernel_height + 1

output_width = image_width - kernel_width + 1

output = np.zeros((output_height, output_width))

completed at 8:42 PM

AISCIENCES

WWW.AISCIENCES.IO

Odemy

Pytorch.ipynb - Colab

Animations of Convolution and

Image Kernels explained visually

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=mbNvxTAdmFg3

Files

sample_data

Convolution Process

```
from google.colab import drive

# Mount Google Drive
drive.mount('/content/drive')
```

Mounted at /content/drive

```
[1] import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

# Function to perform convolution
def convolve2d(image, kernel):
    kernel_height, kernel_width = kernel.shape
    image_height, image_width = image.shape

    output_height = image_height - kernel_height + 1
    output_width = image_width - kernel_width + 1

    output = np.zeros((output_height, output_width))
```

RAM

Disk

80.28 GB available

completed at 8:42 PM

AISCIENCES

Odemy

Pytorch.ipynb - Colab

Animations of Convolution and

Image Kernels explained visually

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=mbNvxTAdmFg3

Files

..

drive

drive data

Convolution Process

27s

```
from google.colab import drive

# Mount Google Drive
drive.mount('/content/drive')
```

Mounted at /content/drive

0s

```
[1] import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

# Function to perform convolution
def convolve2d(image, kernel):
    kernel_height, kernel_width = kernel.shape
    image_height, image_width = image.shape

    output_height = image_height - kernel_height + 1
    output_width = image_width - kernel_width + 1

    output = np.zeros((output_height, output_width))
```

27s completed at 8:43 PM

AISCIENCES

WWW.AISCIENCES.IO

Odeemy

Pytorch.ipynb - Colab

Animations of Convolution and

Image Kernels explained visually

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=IC8-fpW_W2--

RAMDisk

Paused

+ Code + Text

[1]

0s

```
output = np.zeros((output_height, output_width))

for i in range(output_height):
    for j in range(output_width):
        patch = image[i:i+kernel_height, j:j+kernel_width]
        output[i, j] = np.sum(patch * kernel)

return output
```

image_path = r"/content/drive/MyDrive/Dataset/images.jpeg"
image = Image.open(image_path).convert('L') # Convert to grayscale
image = np.array(image)

image = np.random.rand(5, 5)

Define a kernel (e.g., edge detection)
kernel = np.array([
 [1, 0, -1],
 [1, 0, -1],
 [1, 0, -1]
])

Apply convolution
convolved_image = convolve2d(image, kernel)

27s

completed at 8:43 PM

AISCIENCES

Odemy

Pytorch.ipynb - Colab

Animations of Convolution and

Image Kernels explained visually

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=IC8-fpW_W2--

2s

Plot the results

fig, axes = plt.subplots(1, 2, figsize=(15, 5))

axes[0].imshow(image, cmap='gray')

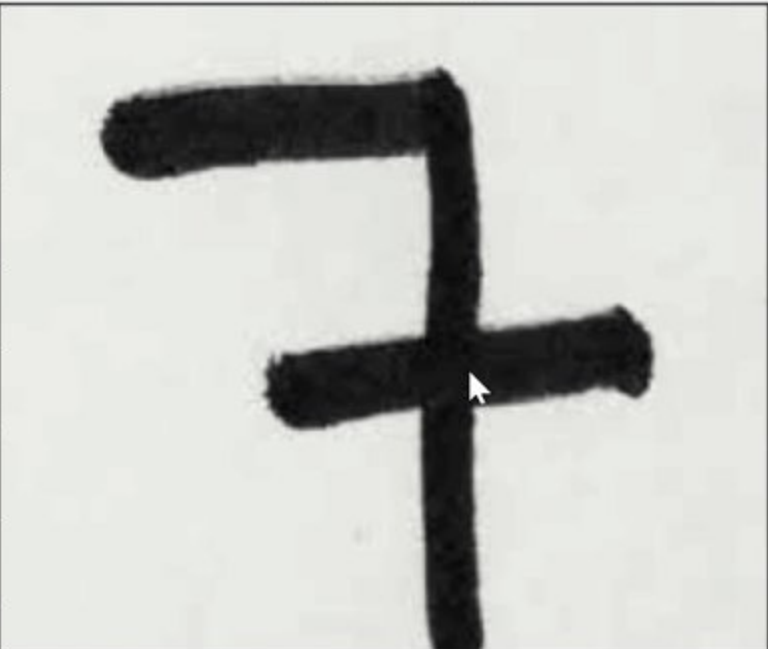
axes[0].set_title('Original Image')

axes[1].imshow(convolved_image, cmap='gray')

axes[1].set_title('Convolved Image')


Text(0.5, 1.0, 'Convolved Image')

Original Image



0 25 50 75 100 125 150 175

Convolved Image



0 25 50 75 100 125 150 175

2s

completed at 8:44 PM

WWW.AISCIENCES.IO

Odemy

Pytorch.ipynb - Colab

Animations of Convolution and

Image Kernels explained visually

colab.research.google.com/drive/1imQs4frEn2sYP0VW7SjxgJwC5ZgNZdpv#scrollTo=IC8-fpW_W2--

+ Code + Text

2s

```
axes[1].imshow(convolved_image, cmap='gray')
axes[1].set_title('Convolved Image')
Text(0.5, 1.0, 'Convolved Image')
```

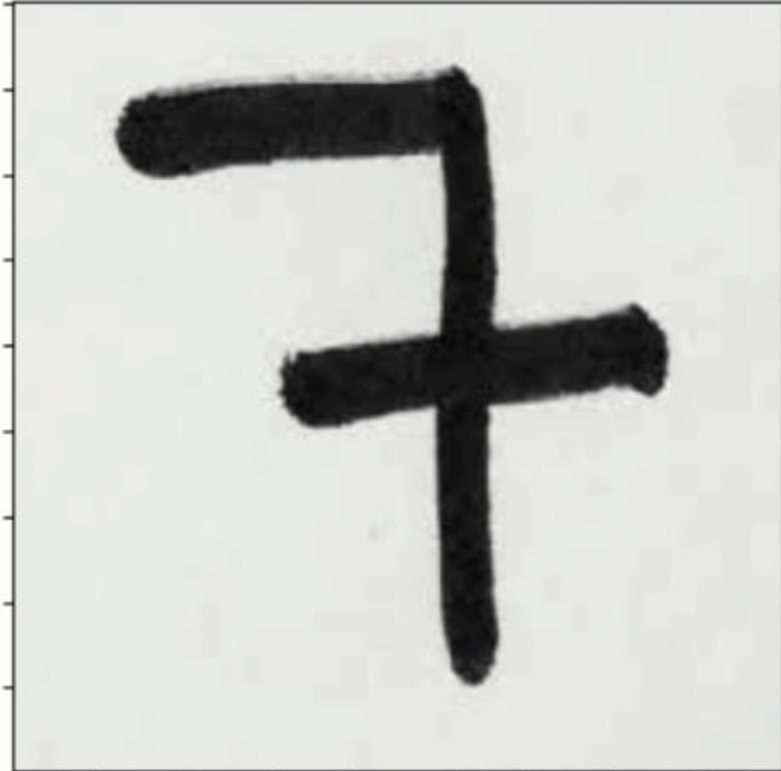
RAM

Disk

Paused

↑ ↓ ↻ ⌨ ⚙ 📄 🗑 ⋮

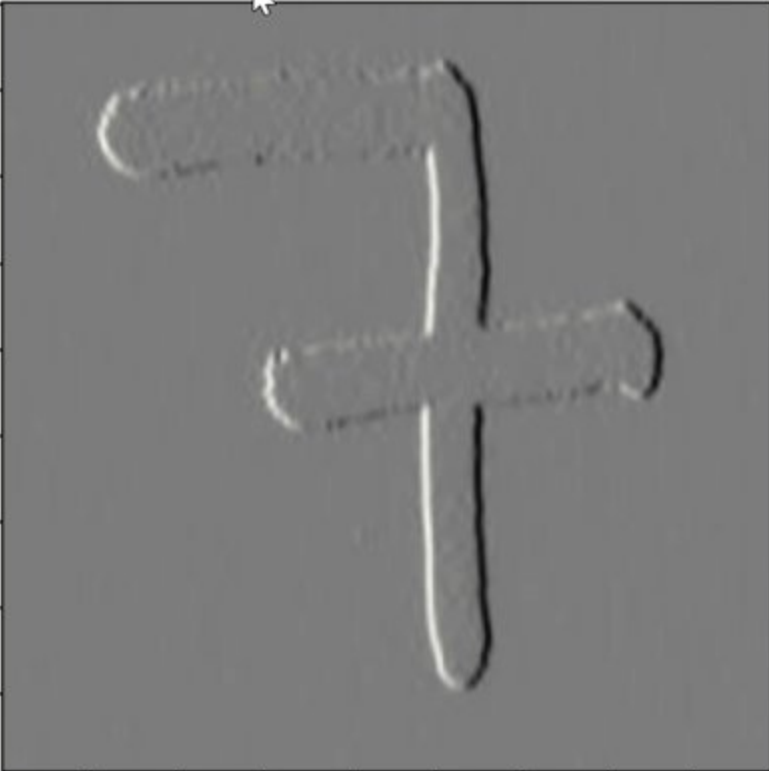
Original Image



0 25 50 75 100 125 150 175 200

0 25 50 75 100 125 150 175 200

Convolved Image



0 25 50 75 100 125 150 175 200

0 25 50 75 100 125 150 175 200

✓ 2s completed at 8:44 PM

WWW.AISCIENCES.IO

AI SCIENCES

Odemy



Machine Learning Lecture

Search this book...

Intro and Overview Machine Learning
Lecture

INTRODUCTION

Basic Concepts of Data Mining and
Machine Learning

CONVENTIONAL ML

K - Nearest Neighbour Classification /
Regression

Bayes- and Naive Bayes Classifier

Linear Regression

Example Linear Regression



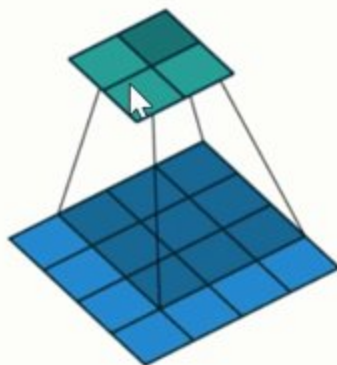
Contents

Convolution

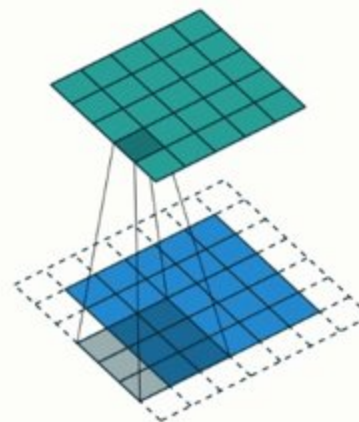
Deconvolution

Convolution

padding = 0, stride = 1



padding = 1, stride = 1



padding = 0, stride = 2



padding = 1, stride = 2



WWW.AISCIENCES.IO



Image Kernels

Explained Visually

[Tweet](#) [Like 442](#) [Share](#)

By [Victor Powell](#)

An image kernel is a small matrix used to apply effects like the ones you might find in Photoshop or Gimp, such as blurring, sharpening, outlining or embossing. They're also used in machine learning for 'feature extraction', a technique for determining the most important portions of an image. In this context the process is referred to more generally as "convolution" (see: [convolutional neural networks](#).)

To see how they work, let's start by inspecting a black and white image. The matrix on the left contains numbers, between 0 and 255, which each correspond to the brightness of one pixel in a picture of a face. The large, granulated picture has been blown up to make it easier to see; the last image is the "real" size.

208 226 247 245 244 253 247 245 138 151 255 255 255 255 255 255 254 257 251 255 254 254 255 255 254 255 252 255 255 254 255 247
244 181 137 244 254 255 254 255 118 103 259 225 155 153 238 193 74 52 88 173 255 254 254 255 255 255 254 255 254 253 244 194
102 154 75 200 240 255 255 255 110 98 94 81 39 44 89 53 44 45 43 54 140 213 255 255 255 255 255 245 187 188 178 225
90 109 98 142 223 255 255 252 117 75 41 35 31 24 25 38 45 44 44 48 91 118 145 234 252 254 255 245 231 245 255 254
87 89 107 136 236 255 255 255 104 25 34 35 29 20 25 34 32 30 32 34 93 85 100 142 231 242 247 240 255 255 255
95 51 45 134 218 251 255 252 51 12 26 33 24 24 45 75 82 76 71 88 98 93 87 80 138 228 238 158 253 245 240 255
79 58 58 75 224 255 255 118 11 27 74 88 91 106 140 162 173 173 173 172 168 137 92 48 78 187 217 238 254 222 233 255
38 43 47 52 147 255 229 98 41 81 129 145 165 165 165 172 178 179 179 179 177 177 172 110 31 82 259 238 255 244 240 255
40 40 33 38 90 245 171 32 85 110 139 145 151 162 171 174 176 179 182 184 187 183 179 182 71 45 187 255 254 255 254 255
37 44 44 31 89 250 188 38 70 129 143 142 153 162 171 175 177 178 182 181 184 188 185 170 120 51 137 255 254 250 254 255
34 45 51 84 118 237 181 53 118 138 140 143 154 164 178 178 174 177 183 188 185 185 183 178 140 88 141 254 252 225 240 255
34 38 52 74 71 188 188 83 131 134 144 155 165 181 173 179 179 179 189 193 190 185 187 182 188 93 145 250 254 214 247 255
32 38 52 94 159 250 128 57 129 138 138 140 151 158 168 168 171 178 180 187 188 185 185 183 183 132 136 242 255 255 254 254
38 32 72 129 212 228 115 85 121 104 102 104 94 103 134 158 170 182 125 108 121 143 145 150 151 104 134 230 255 253 255 251
81 82 118 107 179 247 124 80 101 80 111 119 103 81 94 147 101 178 128 88 125 153 147 181 200 92 100 222 257 187 227 215



Pytorch.ipynb - Colab

Animations of Convolution and

Image Kernels explained visual

setosa.io/ev/image-kernels/

To see how they work, let's start by inspecting a black and white image. The matrix on the left contains numbers, between 0 and 255, which each correspond to the brightness of one pixel in a picture of a face. The large, granulated picture has been blown up to make it easier to see; the last image is the "real" size.

208 205 247 245 244 253 247 245 138 151 255 255 255 255 255 254 257 231 255 254 254 255 255 254 255 252 255 255 254 255 247

244 181 137 244 254 255 254 255 118 133 239 228 155 153 238 133 74 52 88 173 255 254 254 255 255 255 254 255 254 255 244 154

132 154 75 230 242 255 255 255 110 88 84 81 35 44 89 53 44 45 43 54 140 213 253 255 255 255 255 245 187 188 178 223

90 130 98 143 225 255 252 117 76 47 35 31 24 25 38 45 44 44 48 81 118 145 254 252 254 255 248 231 248 255 254

87 89 107 136 238 255 255 255 154 25 34 35 25 25 25 34 32 35 32 34 53 85 130 142 231 242 247 249 255 255 255 255

55 57 45 134 218 251 255 252 57 12 28 33 24 24 48 76 82 78 71 88 58 53 87 93 138 228 238 158 253 248 243 255

79 58 58 75 224 255 255 118 11 27 14 88 91 138 140 162 173 173 173 172 168 137 92 48 78 187 217 238 254 232 233 255

38 43 47 52 147 255 229 58 47 81 125 145 185 185 172 178 179 178 179 177 177 172 150 31 82 239 238 255 244 243 255

40 40 33 38 93 245 171 32 85 110 139 145 151 162 171 114 178 179 182 184 187 183 173 182 71 45 187 255 254 255 254 255

37 44 44 37 89 280 158 38 70 125 143 142 153 162 171 175 177 178 182 181 184 188 183 170 120 57 137 255 254 250 254 255

34 45 51 84 116 237 181 53 118 138 140 143 154 164 178 178 174 177 183 188 189 189 183 178 140 88 141 254 252 225 243 255

34 38 52 74 71 188 158 83 131 134 156 185 185 181 173 179 178 179 180 183 183 185 187 182 158 33 148 250 254 214 247 255

32 35 52 54 159 250 128 57 129 138 138 140 151 158 188 188 171 178 180 187 188 185 185 183 180 132 138 242 255 255 254 254

38 32 72 129 212 228 115 85 121 134 132 134 94 103 134 158 170 182 125 138 121 143 155 180 131 134 134 230 253 253 255 251

81 82 116 107 179 247 124 85 101 80 111 119 103 81 94 147 131 178 128 88 123 153 147 181 200 92 130 222 257 187 227 215

144 118 187 231 210 232 170 67 115 88 78 82 83 86 88 138 132 138 138 81 53 88 141 185 251 57 79 132 245 235 248 248

127 145 149 135 254 213 187 85 133 122 117 133 128 138 110 139 151 157 187 129 127 148 147 171 188 110 121 228 233 180 215 212

87 112 100 79 85 82 85 76 142 148 151 163 138 125 120 149 151 180 183 175 174 183 188 180 208 127 183 239 219 149 158 138

83 83 109 134 129 138 39 78 132 142 165 159 139 111 124 184 158 200 188 182 181 135 200 202 200 143 217 253 243 238 234

89 78 78 113 57 74 43 138 127 140 152 155 125 57 112 160 185 134 174 183 188 188 202 208 208 188 247 254 255 254 254 254

72 44 83 59 48 52 49 74 127 137 146 149 132 103 78 80 134 141 188 185 189 207 204 203 218 193 238 244 251 242 238 243

85 281 89 73 59 80 48 74 117 127 144 181 148 124 126 120 158 187 180 182 189 208 201 205 214 134 174 185 187 188 183 183

85 45 77 83 50 88 43 81 109 127 141 147 113 130 121 145 148 189 181 178 181 201 201 205 202 174 188 189 178 183 188 184

82 78 92 79 54 58 37 47 80 121 132 118 89 78 111 148 183 149 122 124 183 187 187 188 178 149 148 152 155 157 159 188

104 107 122 123 105 79 21 33 88 111 122 120 114 114 147 175 180 138 183 121 170 200 187 185 188 148 145 139 137 141 140 145

117 124 127 133 135 105 21 28 37 88 115 121 128 128 141 142 188 202 212 153 184 188 183 188 134 148 144 149 151 151 147 144

119 118 118 125 128 111 21 29 28 98 130 118 131 140 151 159 188 201 205 192 183 188 149 188 119 144 147 143 140 141 144 148

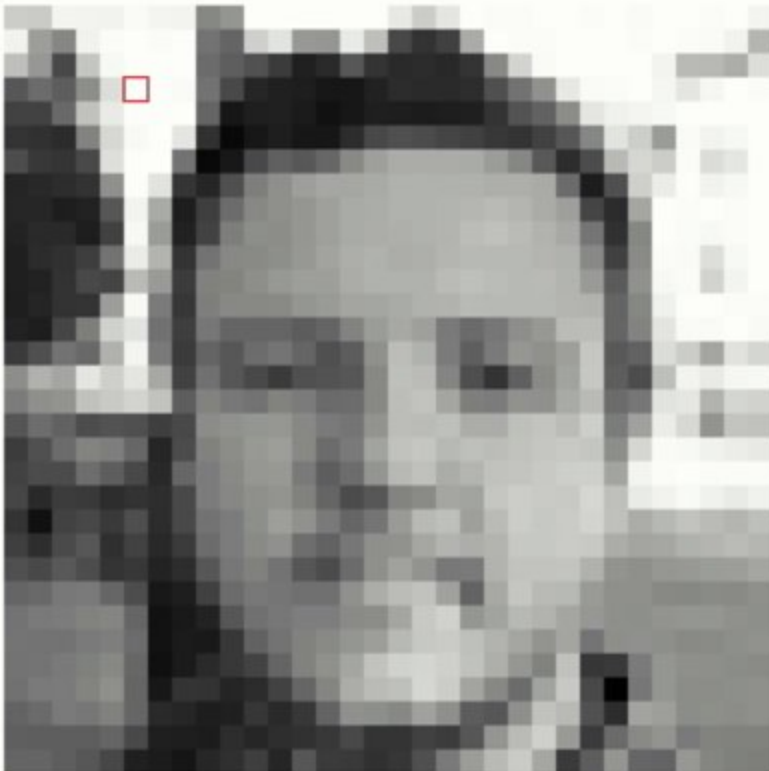
117 118 126 130 139 108 18 29 44 58 70 102 133 147 188 187 212 215 210 185 177 162 133 135 57 58 128 151 145 143 142 141


115 125 128 134 140 102 27 34 52 38 45 89 109 139 179 189 193 218 208 188 139 111 184 203 74 5 121 151 142 142 143 148

101 108 123 121 132 105 44 40 37 36 57 44 58 101 147 144 138 183 145 54 50 145 158 187 94 48 185 185 142 144 142 145

98 97 97 98 104 76 74 33 30 45 47 49 51 58 74 53 55 88 83 89 188 188 209 198 82 138 140 149 125 133 131 131

102 102 97 88 73 36 30 23 42 58 85 47 80 88 59 57 57 82 123 157 187 205 189 82 58 151 135 131 134 136 130 129





Let's walk through applying the following 3x3 **sharpen** kernel to the image of a face from above.

sharpen

AI SCIENCES

WWW.AI SCIENCES.IO

Odemy

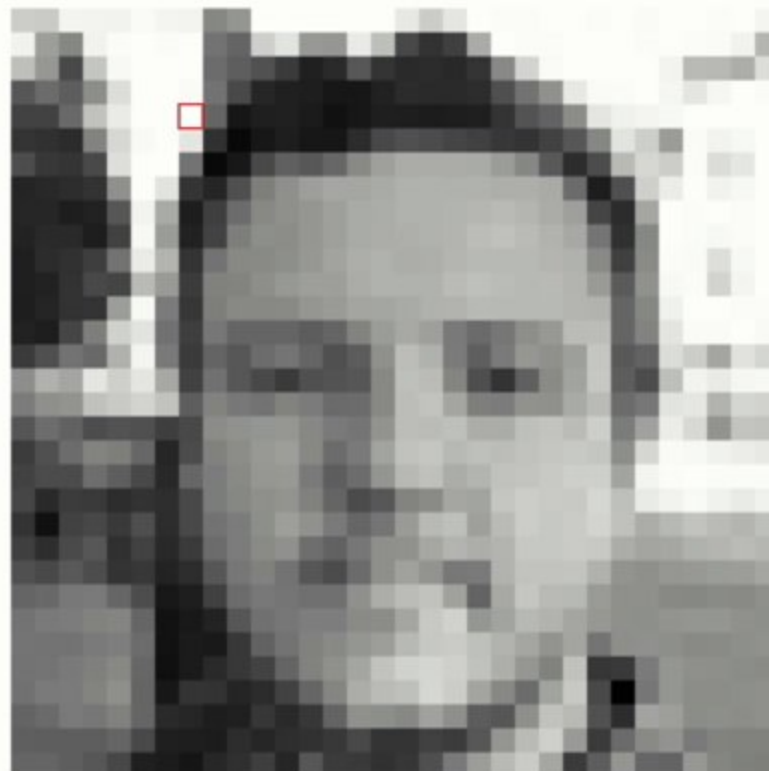
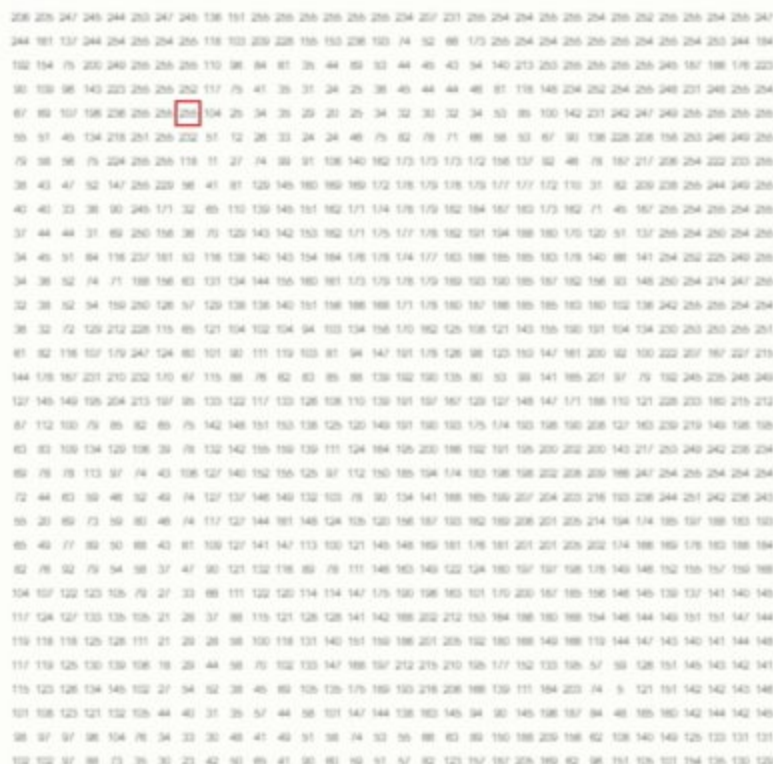
To see how they work, let's start by inspecting a black and white image. The matrix on the left contains numbers, between 0 and 255, which each correspond to the brightness of one pixel in a picture of a face. The large, granulated picture has been blown up to make it easier to see; the last image is the "real" size.



Let's walk through applying the following 3x3 **sharpen** kernel to the image of a face from above.

sharpen

To see how they work, let's start by inspecting a black and white image. The matrix on the left contains numbers, between 0 and 255, which each correspond to the brightness of one pixel in a picture of a face. The large, granulated picture has been blown up to make it easier to see; the last image is the "real" size.



Let's walk through applying the following 3x3 **sharpen** kernel to the image of a face from above.

sharpen

Pytorch.ipynb - Colab

Animations of Convolution and

Image Kernels explained visual


setosa.io/ev/image-kernels/

110 120 130 134 140 102 27 34 32 38 40 69 105 126 176 189 193 218 208 198 139 111 164 203 24 5 121 161 142 142 143 146

101 108 123 121 132 126 44 40 31 26 57 44 58 101 147 144 138 163 146 94 93 145 188 167 94 48 166 180 142 144 142 145

98 97 97 98 104 76 34 33 30 48 41 49 97 98 24 13 55 88 83 89 160 188 198 82 108 140 140 133 131 131

102 102 97 88 73 26 30 23 42 50 89 41 90 87 98 51 57 82 123 167 187 206 166 82 98 101 106 101 104 106 130 129




Let's walk through applying the following 3x3 **sharpen** kernel to the image of a face from above.


sharpen

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

Below, for each 3x3 block of pixels in the image on the left, we multiply each pixel by the corresponding entry of the kernel and then take the sum. That sum becomes a new pixel in the image on the right. Hover over a pixel on either image to see how its value is computed.



$$\begin{pmatrix} ? & 36 & 32 \\ \times 0 & \times -1 & \times 0 \\ + & ? & 61 & 82 \end{pmatrix}$$



WWW.AISCIENCES.IO

AI SCIENCES

Odemy

Pytorch.ipynb - Colab

Animations of Convolution and

Image Kernels explained visual

setosa.io/ev/image-kernels/

Paused

115 120 125 134 140 152 17 34 32 38 40 89 105 126 175 189 193 218 208 198 139 111 164 203 24 5 121 151 142 142 143 146

101 108 123 121 132 125 44 40 31 35 57 44 58 101 147 144 138 163 145 94 93 145 188 157 94 48 165 180 142 144 142 145

98 97 97 98 104 76 34 33 30 48 41 49 97 98 24 93 88 89 83 85 160 188 200 198 82 105 140 140 125 133 131

102 102 97 88 73 35 30 23 42 91 89 41 90 87 94 51 57 82 123 157 187 205 160 82 98 101 105 101 104 105 130 129

Let's walk through applying the following 3x3 **blur** kernel to the image of a face from above.

blur

blur

bottom sobel

emboss

identity

left sobel

outline

right sobel

sharpen

top sobel

0.0625

0.125

0.0625

0.125

0.0625

Below, for each 3x3 block of pixels in the image on the left, we multiply each pixel by the corresponding entry of the kernel and then take the sum. That sum becomes a new pixel in the image on the right. Hover over a pixel on either image to see how its value is computed.

?

36

32

× 0.0625

× 0.125

× 0.0625

?

61

82

× 0.0625

× 0.125

× 0.0625

WWW.AISCIENCES.IO

AI SCIENCES

Odeemy

Pytorch.ipynb - Colab

Animations of Convolution and

Image Kernels explained visual

setosa.io/ev/image-kernels/

Paused

115 120 125 134 140 152 164 174 182 188 195 205 215 225 235 244 254 263 274 281 291 301 311 321 331 342 352 362 370 378 386 393 400 408 415 423 431 438 444 451 458 465 472 479 485 492 500 507 514 521 528 535 542 549 556 563 570 577 584 591 598 605 612 619 626 633 640 647 654 661 668 675 682 689 696 703 710 717 724 731 738 745 752 759 766 773 780 787 794 801 808 815 822 829 836 843 850 857 864 871 878 885 892 899 906 913 920 927 934 941 948 955 962 969 976 983 990 997 1004 1011 1018 1025 1032 1039 1046 1053 1060 1067 1074 1081 1088 1095 1102 1109 1116 1123 1130 1137 1144 1151 1158 1165 1172 1179 1186 1193 1200 1207 1214 1221 1228 1235 1242 1249 1256 1263 1270 1277 1284 1291 1298 1305 1312 1319 1326 1333 1340 1347 1354 1361 1368 1375 1382 1389 1396 1403 1410 1417 1424 1431 1438 1445 1452 1459 1466 1473 1480 1487 1494 1501 1508 1515 1522 1529 1536 1543 1550 1557 1564 1571 1578 1585 1592 1599 1606 1613 1620 1627 1634 1641 1648 1655 1662 1669 1676 1683 1690 1697 1704 1711 1718 1725 1732 1739 1746 1753 1760 1767 1774 1781 1788 1795 1802 1809 1816 1823 1830 1837 1844 1851 1858 1865 1872 1879 1886 1893 1900 1907 1914 1921 1928 1935 1942 1949 1956 1963 1970 1977 1984 1991 1998 2005 2012 2019 2026 2033 2040 2047 2054 2061 2068 2075 2082 2089 2096 2103 2110 2117 2124 2131 2138 2145 2152 2159 2166 2173 2180 2187 2194 2201 2208 2215 2222 2229 2236 2243 2250 2257 2264 2271 2278 2285 2292 2299 2306 2313 2320 2327 2334 2341 2348 2355 2362 2369 2376 2383 2390 2397 2404 2411 2418 2425 2432 2439 2446 2453 2460 2467 2474 2481 2488 2495 2502 2509 2516 2523 2530 2537 2544 2551 2558 2565 2572 2579 2586 2593 2600 2607 2614 2621 2628 2635 2642 2649 2656 2663 2670 2677 2684 2691 2698 2705 2712 2719 2726 2733 2740 2747 2754 2761 2768 2775 2782 2789 2796 2803 2810 2817 2824 2831 2838 2845 2852 2859 2866 2873 2880 2887 2894 2901 2908 2915 2922 2929 2936 2943 2950 2957 2964 2971 2978 2985 2992 2999 3006 3013 3020 3027 3034 3041 3048 3055 3062 3069 3076 3083 3090 3097 3104 3111 3118 3125 3132 3139 3146 3153 3160 3167 3174 3181 3188 3195 3202 3209 3216 3223 3230 3237 3244 3251 3258 3265 3272 3279 3286 3293 3300 3307 3314 3321 3328 3335 3342 3349 3356 3363 3370 3377 3384 3391 3398 3405 3412 3419 3426 3433 3440 3447 3454 3461 3468 3475 3482 3489 3496 3503 3510 3517 3524 3531 3538 3545 3552 3559 3566 3573 3580 3587 3594 3601 3608 3615 3622 3629 3636 3643 3650 3657 3664 3671 3678 3685 3692 3699 3706 3713 3720 3727 3734 3741 3748 3755 3762 3769 3776 3783 3790 3797 3804 3811 3818 3825 3832 3839 3846 3853 3860 3867 3874 3881 3888 3895 3902 3909 3916 3923 3930 3937 3944 3951 3958 3965 3972 3979 3986 3993 4000 4007 4014 4021 4028 4035 4042 4049 4056 4063 4070 4077 4084 4091 4098 4105 4112 4119 4126 4133 4140 4147 4154 4161 4168 4175 4182 4189 4196 4203 4210 4217 4224 4231 4238 4245 4252 4259 4266 4273 4280 4287 4294 4301 4308 4315 4322 4329 4336 4343 4350 4357 4364 4371 4378 4385 4392 4399 4406 4413 4420 4427 4434 4441 4448 4455 4462 4469 4476 4483 4490 4497 4504 4511 4518 4525 4532 4539 4546 4553 4560 4567 4574 4581 4588 4595 4602 4609 4616 4623 4630 4637 4644 4651 4658 4665 4672 4679 4686 4693 4700 4707 4714 4721 4728 4735 4742 4749 4756 4763 4770 4777 4784 4791 4798 4805 4812 4819 4826 4833 4840 4847 4854 4861 4868 4875 4882 4889 4896 4903 4910 4917 4924 4931 4938 4945 4952 4959 4966 4973 4980 4987 4994 5001 5008 5015 5022 5029 5036 5043 5050 5057 5064 5071 5078 5085 5092 5099 5106 5113 5120 5127 5134 5141 5148 5155 5162 5169 5176 5183 5190 5197 5204 5211 5218 5225 5232 5239 5246 5253 5260 5267 5274 5281 5288 5295 5302 5309 5316 5323 5330 5337 5344 5351 5358 5365 5372 5379 5386 5393 5400 5407 5414 5421 5428 5435 5442 5449 5456 5463 5470 5477 5484 5491 5498 5505 5512 5519 5526 5533 5540 5547 5554 5561 5568 5575 5582 5589 5596 5603 5610 5617 5624 5631 5638 5645 5652 5659 5666 5673 5680 5687 5694 5701 5708 5715 5722 5729 5736 5743 5750 5757 5764 5771 5778 5785 5792 5799 5806 5813 5820 5827 5834 5841 5848 5855 5862 5869 5876 5883 5890 5897 5904 5911 5918 5925 5932 5939 5946 5953 5960 5967 5974 5981 5988 5995 6002 6009 6016 6023 6030 6037 6044 6051 6058 6065 6072 6079 6086 6093 6100 6107 6114 6121 6128 6135 6142 6149 6156 6163 6170 6177 6184 6191 6198 6205 6212 6219 6226 6233 6240 6247 6254 6261 6268 6275 6282 6289 6296 6303 6310 6317 6324 6331 6338 6345 6352 6359 6366 6373 6380 6387 6394 6401 6408 6415 6422 6429 6436 6443 6450 6457 6464 6471 6478 6485 6492 6499 6506 6513 6520 6527 6534 6541 6548 6555 6562 6569 6576 6583 6590 6597 6604 6611 6618 6625 6632 6639 6646 6653 6660 6667 6674 6681 6688 6695 6702 6709 6716 6723 6730 6737 6744 6751 6758 6765 6772 6779 6786 6793 6800 6807 6814 6821 6828 6835 6842 6849 6856 6863 6870 6877 6884 6891 6898 6905 6912 6919 6926 6933 6940 6947 6954 6961 6968 6975 6982 6989 6996 7003 7010 7017 7024 7031 7038 7045 7052 7059 7066 7073 7080 7087 7094 7101 7108 7115 7122 7129 7136 7143 7150 7157 7164 7171 7178 7185 7192 7199 7206 7213 7220 7227 7234 7241 7248 7255 7262 7269 7276 7283 7290 7297 7304 7311 7318 7325 7332 7339 7346 7353 7360 7367 7374 7381 7388 7395 7402 7409 7416 7423 7430 7437 7444 7451 7458 7465 7472 7479 7486 7493 7500 7507 7514 7521 7528 7535 7542 7549 7556 7563 7570 7577 7584 7591 7598 7605 7612 7619 7626 7633 7640 7647 7654 7661 7668 7675 7682 7689 7696 7703 7710 7717 7724 7731 7738 7745 7752 7759 7766 7773 7780 7787 7794 7801 7808 7815 7822 7829 7836 7843 7850 7857 7864 7871 7878 7885 7892 7899 7906 7913 7920 7927 7934 7941 7948 7955 7962 7969 7976 7983 7990 7997 8004 8011 8018 8025 8032 8039 8046 8053 8060 8067 8074 8081 8088 8095 8102 8109 8116 8123 8130 8137 8144 8151 8158 8165 8172 8179 8186 8193 8200 8207 8214 8221 8228 8235 8242 8249 8256 8263 8270 8277 8284 8291 8298 8305 8312 8319 8326 8333 8340 8347 8354 8361 8368 8375 8382 8389 8396 8403 8410 8417 8424 8431 8438 8445 8452 8459 8466 8473 8480 8487 8494 8501 8508 8515 8522 8529 8536 8543 8550 8557 8564 8571 8578 8585 8592 8599 8606 8613 8620 8627 8634 8641 8648 8655 8662 8669 8676 8683 8690 8697 8704 8711 8718 8725 8732 8739 8746 8753 8760 8767 8774 8781 8788 8795 8802 8809 8816 8823 8830 8837 8844 8851 8858 8865 8872 8879 8886 8893 8900 8907 8914 8921 8928 8935 8942 8949 8956 8963 8970 8977 8984 8991 8998 9005 9012 9019 9026 9033 9040 9047 9054 9061 9068 9075 9082 9089 9096 9103 9110 9117 9124 9131 9138 9145 9152 9159 9166 9173 9180 9187 9194 9201 9208 9215 9222 9229 9236 9243 9250 9257 9264 9271 9278 9285 9292 9299 9306 9313 9320 9327 9334 9341 9348 9355 9362 9369 9376 9383 9390 9397 9404 9411 9418 9425 9432 9439 9446 9453 9460 9467 9474 9481 9488 9495 9502 9509 9516 9523 9530 9537 9544 9551 9558 9565 9572 9579 9586 9593 9600 9607 9614 9621 9628 9635 9642 9649 9656 9663 9670 9677 9684 9691 9698 9705 9712 9719 9726 9733 9740 9747 9754 9761 9768 9775 9782 9789 9796 9803 9810 9817 9824 9831 9838 9845 9852 9859 9866 9873 9880 9887 9894 9901 9908 9915 9922 9929 9936 9943 9950 9957 9964 9971 9978 9985 9992 9999 10000

Let's walk through applying the following 3x3 **top sobel** kernel to the image of a face from above.

top sobel

1

2

1

0

0

0

-1

-2

-1

Below, for each 3x3 block of pixels in the image on the left, we multiply each pixel by the corresponding entry of the kernel and then take the sum. That sum becomes a new pixel in the image on the right. Hover over a pixel on either image to see how its value is computed.

?

36

32

× 1

× 2

× 1

+

?

61

82

WWW.AISCIENTES.IO

AI SCIENCES

Odeemy

Pytorch.ipynb - Colab


Animations of Convolution and

Image Kernels explained visual

setosa.io/ev/image-kernels/

Paused

Below, for each 3x3 block of pixels in the image on the left, we multiply each pixel by the corresponding entry of the kernel and then take the sum. That sum becomes a new pixel in the image on the right. Hover over a pixel on either image to see how its value is computed.



input image

142

+ 141

+ ?

× 1

× 2

× 1

+

143

+ 146

+ ?

× 0

× 0

× 0

+

142

+ 145

+ ?

× -1

× -1

× -1

blur

bottom sobel

emboss

identity

left sobel

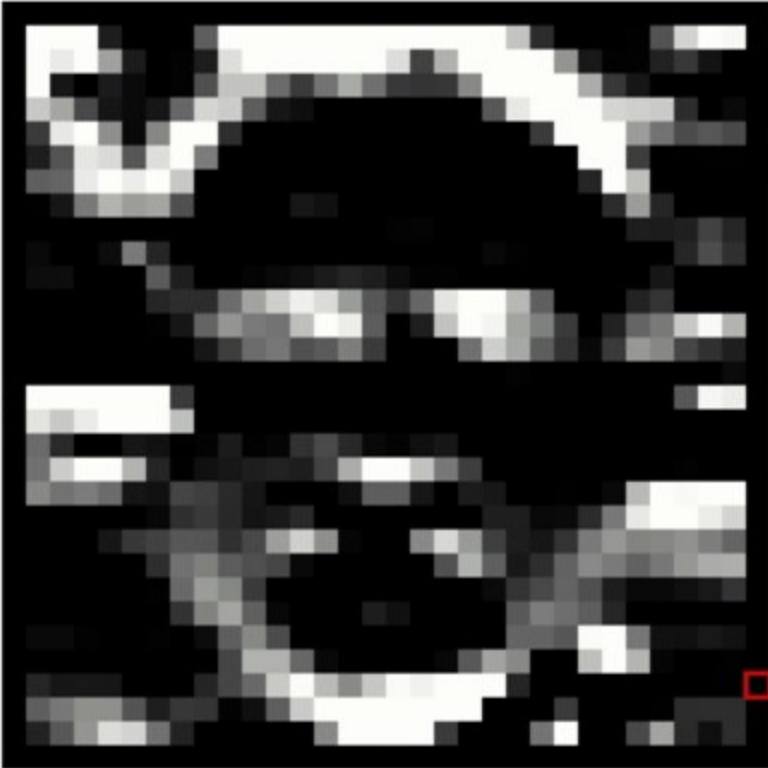
outline

right sobel

sharpen

top sobel

top sobel



output image

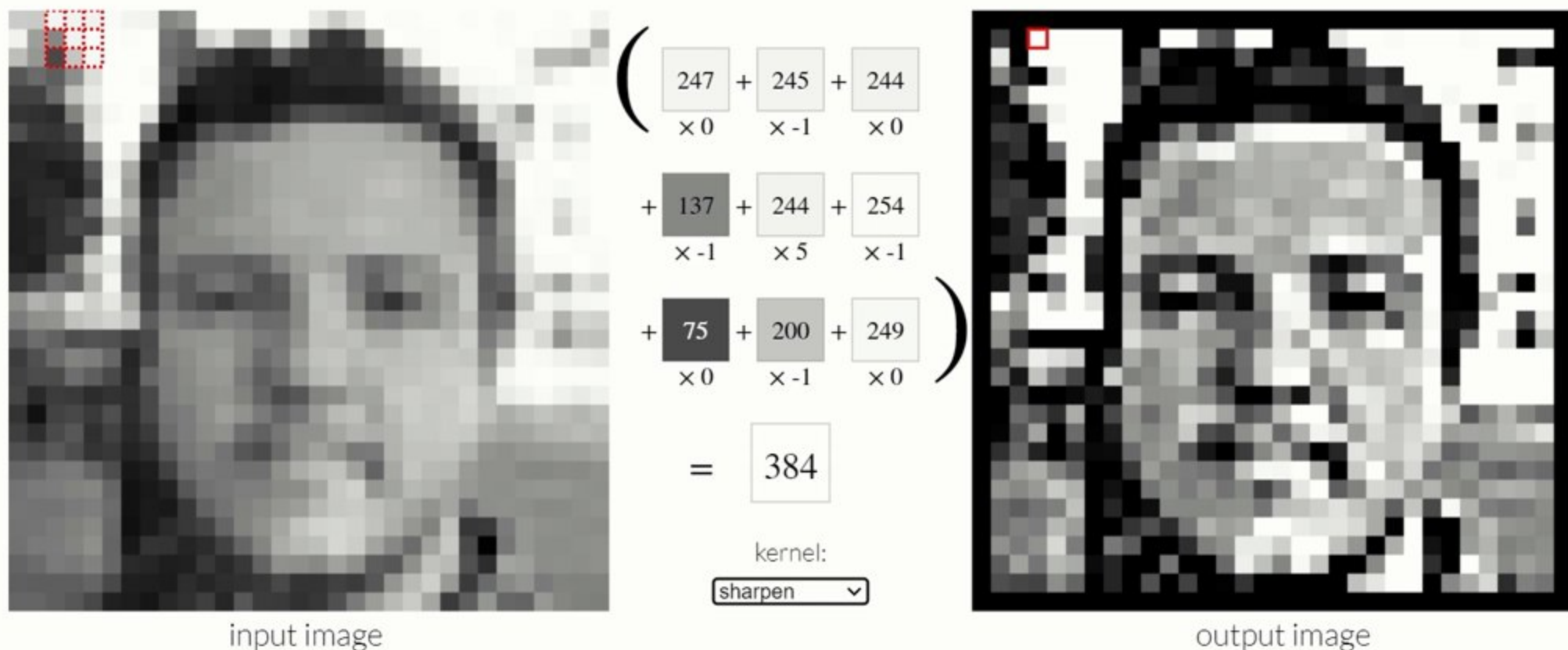
One subtlety of this process is what to do along the edges of the image. For example, the top left corner of the input image only has three neighbors. One way to fix this is to extend the edge values out by one in the original image while keeping our new image the same size. In this demo, we've instead ignored those values by making them black.

WWW.AISCIENCES.IO

AI SCIENCES

@edemy

Below, for each 3x3 block of pixels in the image on the left, we multiply each pixel by the corresponding entry of the kernel and then take the sum. That sum becomes a new pixel in the image on the right. Hover over a pixel on either image to see how its value is computed.



One subtlety of this process is what to do along the edges of the image. For example, the top left corner of the input image only has three neighbors. One way to fix this is to extend the edge values out by one in the original image while keeping our new image the same size. In this demo, we've instead ignored those values by making them black.

Pytorch.ipynb - Colab


Animations of Convolution and

Image Kernels explained visual

setosa.io/ev/image-kernels/

Paused

Below, for each 3x3 block of pixels in the image on the left, we multiply each pixel by the corresponding entry of the kernel and then take the sum. That sum becomes a new pixel in the image on the right. Hover over a pixel on either image to see how its value is computed.



61

109

127

× 0

× -1

× 0

+

+

+

47

90

121

× -1

× 5

× -1

+

+

+

33

66

111

× 0

× -1


× 0

=

107

kernel:

sharpen



output image

One subtlety of this process is what to do along the edges of the image. For example, the top left corner of the input image only has three neighbors. One way to fix this is to extend the edge values out by one in the original image while keeping our new image the same size. In this demo, we've instead ignored those values by making them black.

WWW.AISCIENCES.IO

AI SCIENCES

Odemy

same size. In this demo, we've instead ignored those values by making them black.

Here's a playground where you can select different kernel matrices and see how they effect the original image or build your own kernel. You can also upload your own image or use live video if your browser supports it.

Choose File Cancer (1).png Live video

0	-1	0
-1		-1
0		0

blur

bottom sobel

custom

emboss

identity

left sobel

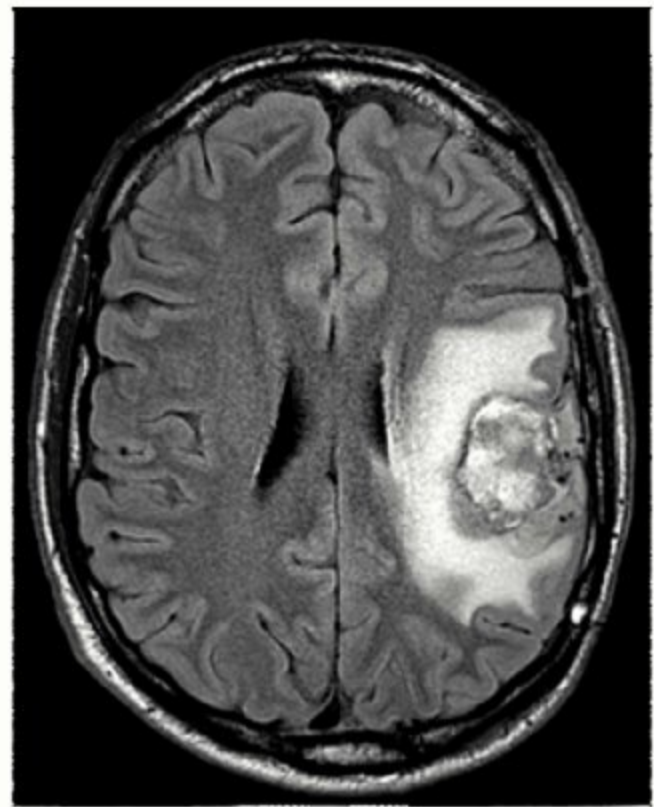
outline

right sobel

sharpen

top sobel

sharpen



The **sharpen** kernel emphasizes differences in adjacent pixel values. This makes the image look more vivid.