Projeto1_parte1

March 9, 2023

0.1 Importando Libs Utilizadas

[ ]:
```
pip install Pillow
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: Pillow in /usr/local/lib/python3.8/dist-packages
(8.4.0)
```

[ ]:
```
import PIL
print(f'Versão Pillow: {PIL.__version__}')
```

```
Versão Pillow: 8.4.0
```

[ ]:
```
pip install opencv-python
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: opencv-python in /usr/local/lib/python3.8/dist-
packages (4.6.0.66)
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.8/dist-
packages (from opencv-python) (1.22.4)
```

[ ]:
```
import cv2
print(cv2.__version__);
```

```
4.6.0
```

[ ]:
```
pip install matplotlib
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: matplotlib in /usr/local/lib/python3.8/dist-
packages (3.5.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-
packages (from matplotlib) (0.11.0)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.8/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.8/dist-
```

```
packages (from matplotlib) (23.0)
Requirement already satisfied: pyparsing>=2.2.1 in
/usr/local/lib/python3.8/dist-packages (from matplotlib) (3.0.9)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.8/dist-
packages (from matplotlib) (8.4.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.8/dist-
packages (from matplotlib) (1.22.4)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.8/dist-packages (from matplotlib) (4.38.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.8/dist-packages (from matplotlib) (1.4.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-
packages (from python-dateutil>=2.7->matplotlib) (1.15.0)
```

[ ]:
```
import matplotlib;
print(matplotlib.__version__);
```

```
3.5.3
```

[ ]:
```
pip install numpy
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages
(1.22.4)
```

[ ]:
```
import numpy as np;
print(np.__version__);
```

```
1.22.4
```

0.2 Carregando Imagem a ser tratada em tons de cinza

[ ]:
```
from PIL import Image

# Carregando a imagem
imagem = Image.open('images/morumbi.png')

# Transformando em uma imagem monocromática
img_escala_de_cinza = imagem.convert('L')

# Salvando a Imagem transformada
img_escala_de_cinza.save('images/morumbi-cinza.png')

# Apresentando a imagem
img_escala_de_cinza.show()
```

0.3 Carregando a Imagem monocromática para ser utilizada nos próximos passos

[ ]:
```
from matplotlib import pyplot as plt
import matplotlib.image as mpimg

# Carrega a imagem como um array de pixels
data = mpimg.imread('images/morumbi-cinza.png')
print(data.shape)
# Imprime os arrays de dados
img_escala_de_cinza.show()
print(data)
```

```
(566, 800)
```


```
[[0.7764706  0.7764706  0.7764706  … 0.94509804 0.94509804 0.94509804]
 [0.7764706  0.7764706  0.78039217 … 0.9607843  0.9607843  0.9607843 ]
 [0.78039217 0.78039217 0.78039217 … 0.95686275 0.95686275 0.9607843 ]
 …
 [0.7058824  0.7019608  0.69803923 … 0.32156864 0.2784314  0.25490198]
 [0.7019608  0.7019608  0.69803923 … 0.3254902  0.31764707 0.30980393]
 [0.67058825 0.6862745  0.69411767 … 0.27450982 0.30980393 0.32156864]]
```

0.4 Função Utilizada para ler o arquivo que contém a máscara genérica

A função recebe o nome do arquivo da máscara e retorna a matriz numérica para ser aplicada nas convoluções

[ ]:
```
def readMask(filename):
  with open(filename) as f:
    w, h = [float(x) for x in next(f).split()]
    array = [[float(x) for x in line.split()] for line in f]
  return array

mask = readMask("mask.txt")
print(mask)
```

```
[[0.3333, 0.3333, 0.3333], [0.3333, 0.3333, 0.3333], [0.3333, 0.3333, 0.3333]]
```

0.5 Função de Convolução sem extensão por 0

A função recebe uma imagem de ordem MxN e uma máscara também MxN e retorna o mapa de ativação resultante da convolução espacial

É feito uma limitação do percorrimento da máscara na imagem de entrada, pois como não tem extensão por 0, parte da máscara em momentos do loop iria enxergar fora da imagem.

[ ]:
```
def convolution_without_extension(image, mask):

  # Array que vai guardar a matriz resultante
  resultImage = np.array([])

  # Extraindo Dimensões da imagem de entrada
  lines, columns = image.shape
  print(lines, columns)

  # Extraindo Dimensões da máscara
  maskLin, maskCol = mask.shape

  # Percorrendo as linhas da imagem de entrada limitadamente
  for i in range(0,lines - (maskLin -1)):

    # Array que acumula a linha que vai ser inserida na matriz resultante
    newLine = np.array([])

    # Percorrendo as colunas da imagem de entrada limitadamente
    for j in range(0, columns - (maskCol -1)):

      # Pedaço da imagem que esstá sendo enxergado pela máscara em cada momento␣
 ↪da convolução
      subImag = image[i:i+maskLin, j:j+maskCol]

      # Produto Interno do pedaço da imagem enxergado com a máscara
      product = np.sum(np.multiply(subImag, mask))

      # Acumula o resultado do produto na linha
      newLine = np.append(newLine, product)

    # Acumulando os resultados na matriz resultante
    resultImage = np.append(resultImage, newLine)

  # Tranformando o array resultante na mztriz resultante
  return resultImage.reshape(lines - (maskLin - 1), columns - (maskCol - 1))

result = convolution_without_extension(data, np.array(mask))
print(result)
```

```
566 800
```

```
[[2.3344071  2.33702122 2.34094239 … 2.86245883 2.86245883 2.86376589]
 [2.33571416 2.33571416 2.33702122 … 2.86899413 2.86899413 2.8729153 ]
 [2.3344071  2.33179298 2.33048592 … 2.86899413 2.86899413 2.87422236]
 …
 [2.0808377  2.10697888 2.13181299 … 0.89272121 0.83782474 0.79992004]
 [2.09260123 2.09782947 2.10044358 … 0.93193298 0.9057918  0.86265886]
 [2.08606594 2.08475888 2.07560947 … 0.87834356 0.90709885 0.90448474]]
```

0.6 Função de Convolução com extensão por 0

A função recebe uma imagem de ordem MxN e uma máscara também MxN e retorna o mapa de ativação resultante da convolução espacial

É criada uma nova imagem à ser convolucionada. Sendo adicionadas linhas e colunas com zeros baseada nas dimensões da máscara. Essa quantidade de linhas e colunas é obtida através do seguinte cálculo: - linhas: (M // 2) * 2 (Onde M é as linhas da máscara e a operação // é o arredondamento da divisão para baixo)

- colunas: (N // 2) * 2 (Onde N é as colunas da máscara e a operação // é o arredondamento da divisão para baixo)

É feito uma limitação do percorrimento da máscara não mais na imagem de entrada, e sim na imagem adicionada os zeros, mantendo assim a ordem da mztriz original na matriz resultante.

[ ]:
```python
def convolution_with_extension(image, mask):

  # Dimensões da imagem e da máscara
  lines, columns = image.shape
  maskLin, maskCol = mask.shape

  # Quantidade de linhas e colunas à ser adicionada
  linesWithZero = maskLin // 2
  columnsWithZero =  maskCol // 2

  # Imagem à ser processadas com os zeros adicionados
  imageWithExt = np.array([])

  # Adicionando as linhas iniciais com 0
  for i in range(linesWithZero):
    imageWithExt = np.append(imageWithExt, np.zeros(shape=(1, columns +␣
↪columnsWithZero * 2)))

  for i in range(0, lines):

    # Coloca os zeros das primeiras colunas de cada linha
    imageWithExt = np.append(imageWithExt, np.zeros(shape=(1, columnsWithZero)))

    # Coloca cada linha completa da matriz de entrada
    imageWithExt = np.append(imageWithExt, image[i])
```

```
    # Coloca os zeros das últimas colunas de cada linha
    imageWithExt = np.append(imageWithExt, np.zeros(shape=(1, columnsWithZero)))

  # Adicionando as linhas finais com 0
  for i in range(linesWithZero):
    imageWithExt = np.append(imageWithExt, np.zeros(shape=(1, columns +␣
 ↪columnsWithZero * 2)))

  # Transformando o array em matriz
  imageWithExt = imageWithExt.reshape(lines + linesWithZero*2, columns +␣
 ↪columnsWithZero * 2)

  # Extraindo Dimensões da imagem a ser processada
  linesWithExt, columnsWithExt = imageWithExt.shape

  # Array que vai guardar a matriz resultante
  resultImage = np.array([])

  # Percorrendo limitadamente a matriz extendida por 0
  for i in range(0,linesWithExt - (maskLin -1)):

    # Array que acumula a linha que vai ser inserida na matriz resultante
    newLine = np.array([])

    for j in range(0, columnsWithExt - (maskCol -1)):

      # Pedaço da imagem que esstá sendo exergado pela máscara em cada momento␣
 ↪da convolução
      subImag = imageWithExt[i:i+maskLin, j:j+maskCol]

      # Produto Interno do pedaço da imagem exergado com a máscara
      product = np.sum(np.multiply(subImag, mask))

      # Acumula o resultado do produto na linha
      newLine = np.append(newLine, product)

    # Acumulando os resultados na matriz resultante
    resultImage = np.append(resultImage, newLine)

  # Tranformando o array resultante na mztriz resultante
  return resultImage.reshape(linesWithExt - (maskLin - 1), columnsWithExt -␣
 ↪(maskCol - 1))

result = convolution_with_extension(data, np.array(mask))
print(result)
```

```
[[1.03519061 1.55409297 1.55670709 … 1.90569177 1.90569177 1.27046118]
 [1.55540003 2.3344071  2.33702122 … 2.86245883 2.86376589 1.90961295]
 [1.55670709 2.33571416 2.33571416 … 2.86899413 2.8729153  1.91614824]
 …
 [1.39332474 2.09260123 2.09782947 … 0.9057918  0.86265886 0.56072826]
 [1.38940357 2.08606594 2.08475888 … 0.90709885 0.90448474 0.59732591]
 [0.92016944 1.38417533 1.38548239 … 0.58948355 0.61954591 0.4195659 ]]
```

0.7 Função Para exibição do mapa de ativação

Após a convolução alguns valores podem ter ficado negativo e com valor absoluto maior que 1 ou 255, impossibilitando a exibição da imagem resultante.

Portanto, nessa função é feito um tratamento para evitar o problema citado acima.

A entrada da função é o mapa de ativação resultante da convolução, e a saída é a exibição da imagem.

[ ]:
```
def showActivationMap(activationMap):
  biggest = 0

  # Percorre toda a matriz procurando o o maior valor para fazer a escala
  for i in range(0, activationMap.shape[0]):
    for j in range(0, activationMap.shape[1]):

      # Transforma todos os valores em positivo
      activationMap[i][j] = abs(activationMap[i][j])

      if(activationMap[i][j] > biggest):
        biggest = activationMap[i][j]

  # Depois de achado o maior valor é feito a escala baseado nele, e depois␣
 ↪multiplicado por 255 para conseguir montar e exibir a imagem
  for i in range(0, activationMap.shape[0]):
    for j in range(0, activationMap.shape[1]):
      activationMap[i][j] /= biggest
      activationMap[i][j] *= 255;

  im = Image.fromarray(activationMap)
  im.show()

showActivationMap(result)
```


LG