```
import torch
import torch.nn as nn
import numpy as np
import tensorflow as tf
from torch.utils.data import DataLoader
import torchvision
import torchvision.transforms as transforms
import matplotlib.pyplot as plt
from PIL import Image
```

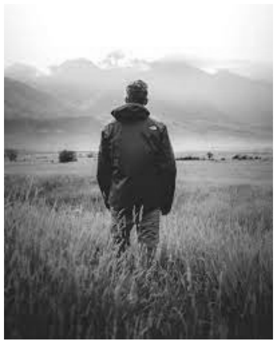
Image Operations

Original Image import

```
image=Image.open('image.jpg')
image.show()
image_arr_BGR=np.array(image)
```

Converting to Greyscale

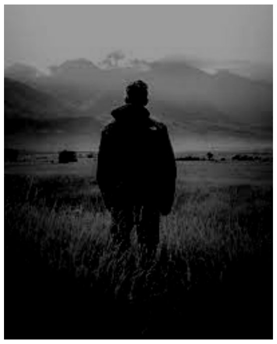
```
# gray=(image_arr_BGR[:,:,0]+image_arr_BGR[:,:,1]+image_arr_BGR[:,:,2])/3
image_arr_Grey=np.empty((251,201))
for i in range(251):
  for j in range(201):
    image_arr_Grey[i][j]=0.299*(image_arr_BGR[i][j][0])+0.587*(image_arr_BGR[i][j][1])+0.114*(image_arr_BGR[i][j][2]
Grey_image=Image.fromarray(image_arr_Grey)
Grey_image.show()
```



Brightness Decreased

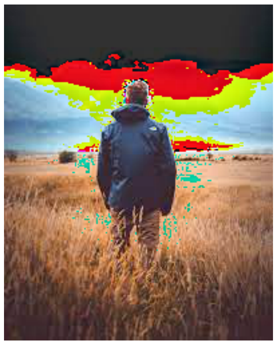
```
#Downscaling pixel values
BGR_arr_low=(image_arr_BGR[:,:,:]*0.4).astype(np.uint8)
BGR_image=Image.fromarray(BGR_arr_low)
BGR_image.show()
```

```
#Decreasing Pixel Values
image_arr_Grey[:,:]-=100
image_arr_Grey=np.clip(image_arr_Grey,0,255)
Grey_image=Image.fromarray(image_arr_Grey)
Grey_image.show()
```



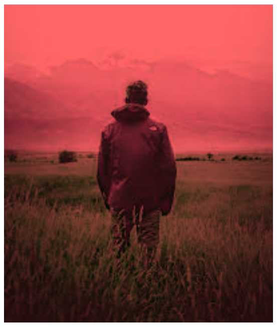
Contrast Operation

```
contrast_value=1.2
image_arr_contrast=np.empty((251,201,3))
#Applying contrast formula
image_arr_contrast=((image_arr_BGR[:,:,:]-0.5)*contrast_value+0.5).astype(np.uint8)
image_arr_contrast=np.clip(image_arr_contrast,0,255)
Contrast_image=Image.fromarray(image_arr_contrast)
Contrast_image.show()
```



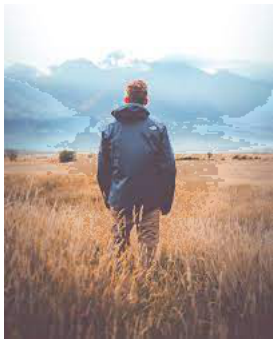
Noise operation

```
img_arr_noise=np.empty((251,201,3))
for i in range(251):
  for j in range(201):
    img_arr_noise[i][j][0]=image_arr_BGR[i][j][0]+np.random.randint(0,2)
    img_arr_noise[i][j][0]=image_arr_BGR[i][j][1]+np.random.randint(0,2)
    img_arr_noise[i][j][0]=image_arr_BGR[i][j][2]+np.random.randint(0,2)
img_arr_noise=np.clip(img_arr_noise,None,255)
img_noise=Image.fromarray(img_arr_noise.astype(np.uint8))
img_noise.show()
```



Brightness increased

```
image_arr_bright=np.empty((251,201,3))
for i in range(251):
  for j in range(201):
    if (image_arr_BGR[i][j][0]>200) or (image_arr_BGR[i][j][1]>200) or (image_arr_BGR[i][j][2]>200):
      image_arr_bright[i][j][0]=image_arr_BGR[i][j][0].astype(np.uint8)
      image_arr_bright[i][j][1]=image_arr_BGR[i][j][1].astype(np.uint8)
      image_arr_bright[i][j][2]=image_arr_BGR[i][j][2].astype(np.uint8)
    else:
      image_arr_bright[i][j][0]=image_arr_BGR[i][j][0].astype(np.uint8)+40
      image_arr_bright[i][j][1]=image_arr_BGR[i][j][1].astype(np.uint8)+40
      image_arr_bright[i][j][2]=image_arr_BGR[i][j][2].astype(np.uint8)+40
image_arr_bright=np.clip(image_arr_bright,0,255).astype(np.uint8)
BGR_image_b=Image.fromarray(image_arr_bright)
BGR_image_b.show()
```



Binary Image operation

```
img_arr_binary=np.empty((251,201))
threshold=100
for i in range(251):
  for j in range(201):
    for k in range(3):
      if(image_arr_BGR[i][j][k]>threshold):
        img_arr_binary[i][j]=255
      else:
        img_arr_binary[i][j]=0
img_binary=Image.fromarray(img_arr_binary)
img_binary.show()
```

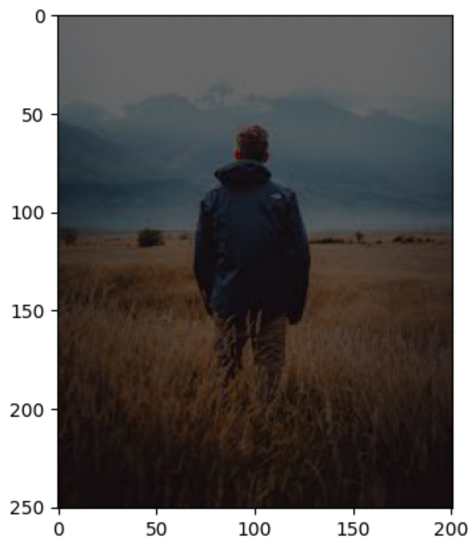
Kernel Operations

[redacted]

```
def convolutionBGR(img, kernel=None):
  '''
    convolution operation for applying a kernel on image
  '''
  if(kernel is None):
    assert('Pass the kernel')
  # kernel=kernel/255
  img = img/255
  kernel_size = kernel.shape[0]
  img_h, img_w, img_c = img.shape
  padding = (kernel_size - 1) // 2
  out_h, out_w, out_c = img_h + (2 * padding), img_w + (2 * padding), img_c
  padded_img = np.zeros(shape=(out_h, out_w, out_c))
  output_img = np.zeros(shape=(img_h, img_w, img_c))

  for i in range(img_h):
    for j in range(img_w):
      padded_img[(i + padding), (j + padding), :] = padded_img[(i + padding), (j + padding), :] + img[i, j]

  for i in range(img_h):
    for j in range(img_w):
      for k in range(img_c):
        output_img[i, j, k] = np.sum(padded_img[i:(i + kernel_size), j:(j + kernel_size), k] * kernel)

  return output_img


def convolutionGrey(img, kernel=None):
  '''
    convolution operation for applying a kernel on image
  '''
  if(kernel is None):
    assert('Pass the kernel')
  # kernel=kernel/255
  img = img/255
  kernel_size = kernel.shape[0]
  img_h, img_w = img.shape
  padding = (kernel_size - 1) // 2
  out_h, out_w = img_h + (2 * padding), img_w + (2 * padding)
  padded_img = np.zeros(shape=(out_h, out_w))
  output_img = np.zeros(shape=(img_h, img_w))

  for i in range(img_h):
    for j in range(img_w):
      padded_img[(i + padding), (j + padding)] = padded_img[(i + padding), (j + padding)] + img[i, j]

  for i in range(img_h):
    for j in range(img_w):
        output_img[i, j] = np.sum(padded_img[i:(i + kernel_size), j:(j + kernel_size)] * kernel)

  return output_img


kernel=np.array([
    [0,0,0],
    [0,0.4,0],
    [0,0,0]
    ])
image_arr_convolved = convolutionBGR(image_arr_BGR,kernel)
plt.imshow(image_arr_convolved)
```

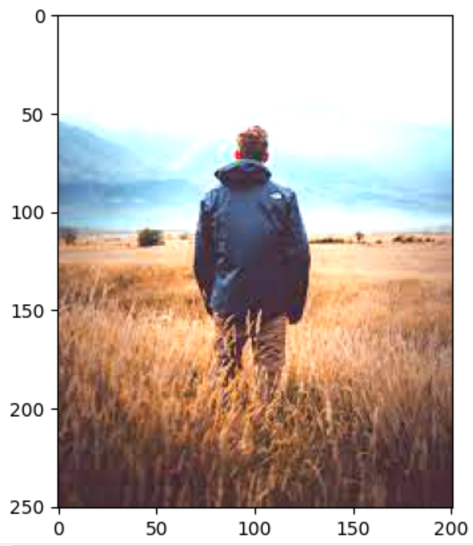
```
# image_convolved=Image.fromarray(image_arr_convolved)
# image_convolved.show()
```

```
<matplotlib.image.AxesImage at 0x7f98497e4490>
```



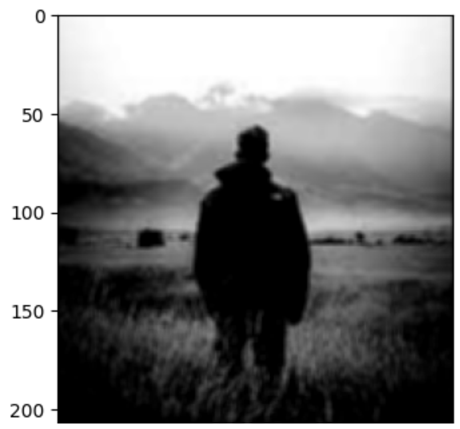
```
kernel=np.array([
    [0,0,0],
    [0,1.4,0],
    [0,0,0]
    ])
image_arr_convolved = convolutionBGR(image_arr_BGR,kernel)
plt.imshow(image_arr_convolved)
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RG
<matplotlib.image.AxesImage at 0x7f984951e080>
```



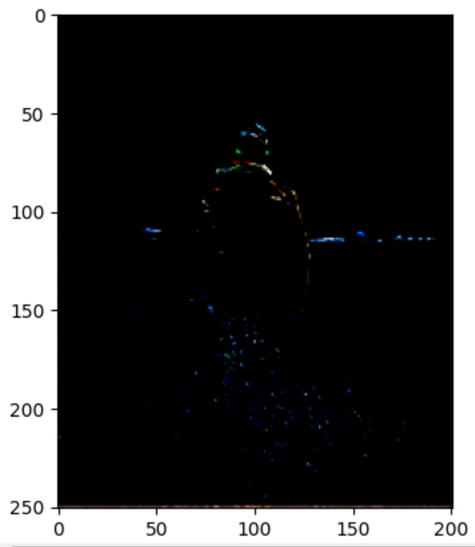
```
kernel=np.array([
    [1,1,1],
    [1,1,1],
    [1,1,1]
    ])
image_arr_convolved = convolutionGrey(image_arr_Grey,kernel)
plt.imshow(image_arr_convolved, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f98494f7850>
```



```
kernel = np.random.normal(loc=0, scale=2, size=(3, 3))
image_arr_convolved = convolutionBGR(image_arr_BGR,kernel)
plt.imshow(image_arr_convolved, cmap='gray')
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RG
<matplotlib.image.AxesImage at 0x7f9849307d90>
```

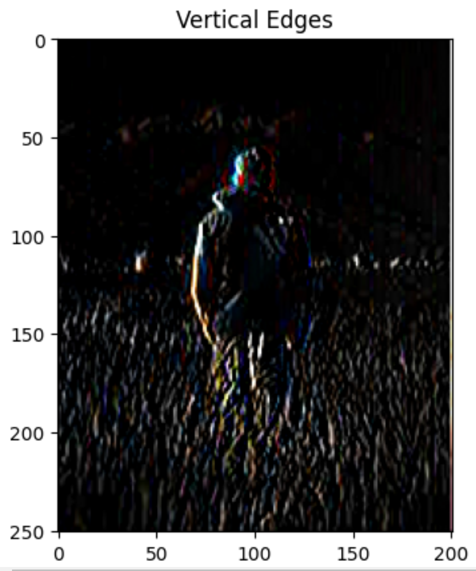


```
kernel = np.array([[1,1,1],[0,0,0],[-1,-1,-1]])
image_arr_convolved = convolutionBGR(image_arr_BGR,kernel)
plt.imshow(image_arr_convolved, cmap='gray')
plt.title('Horizontal Edges')
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RG
Text(0.5, 1.0, 'Horizontal Edges')
```

```
kernel = np.array([[1,0,-1],[1,0,-1],[1,0,-1]])
image_arr_convolved = convolutionBGR(image_arr_BGR,kernel)
plt.imshow(image_arr_convolved, cmap='gray')
plt.title('Vertical Edges')
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RG
Text(0.5, 1.0, 'Vertical Edges')
```



✓ 1s completed at 10:33 PM