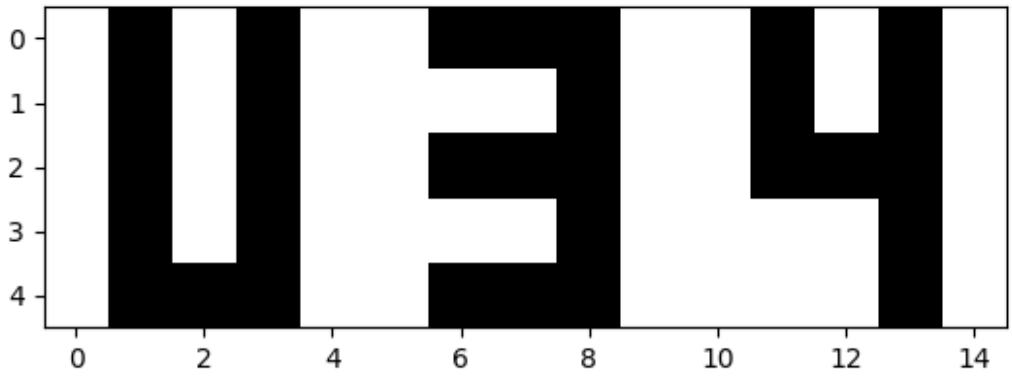
```
import numpy as np
import matplotlib.pyplot as plt

import cv2
```

Task 1

```
Letter_S = np.array(
  [
    [255, 0, 255, 0, 255,     255,   0,   0, 0, 255,     255,   0,
255, 0, 255],
    [255, 0, 255, 0, 255,     255, 255, 255, 0, 255,     255,   0,
255, 0, 255],
    [255, 0, 255, 0, 255,     255,   0,   0, 0, 255,     255,   0,
0, 0, 255],
    [255, 0, 255, 0, 255,     255, 255, 255, 0, 255,     255, 255,
255, 0, 255],
    [255, 0,   0, 0, 255,     255,   0,   0, 0, 255,     255, 255,
255, 0, 255]
  ]
)

plt.imshow(Letter_S,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7e04f36c8220>
```

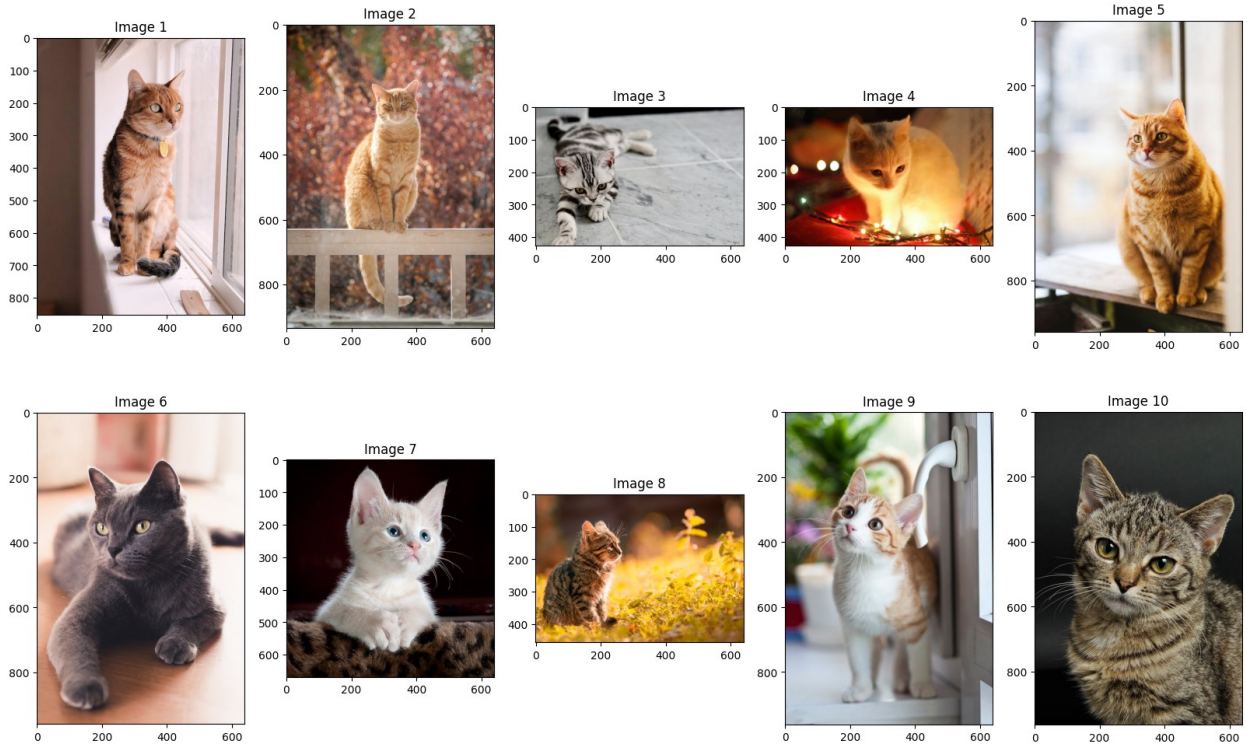


Task 2

```
# All transformation functions - Crop, Flip, Rotate, Resize,
Translate, Shear, Stretch

def crop(image, y, x, h, w):
  return image[y:y+h, x:x+w]
```

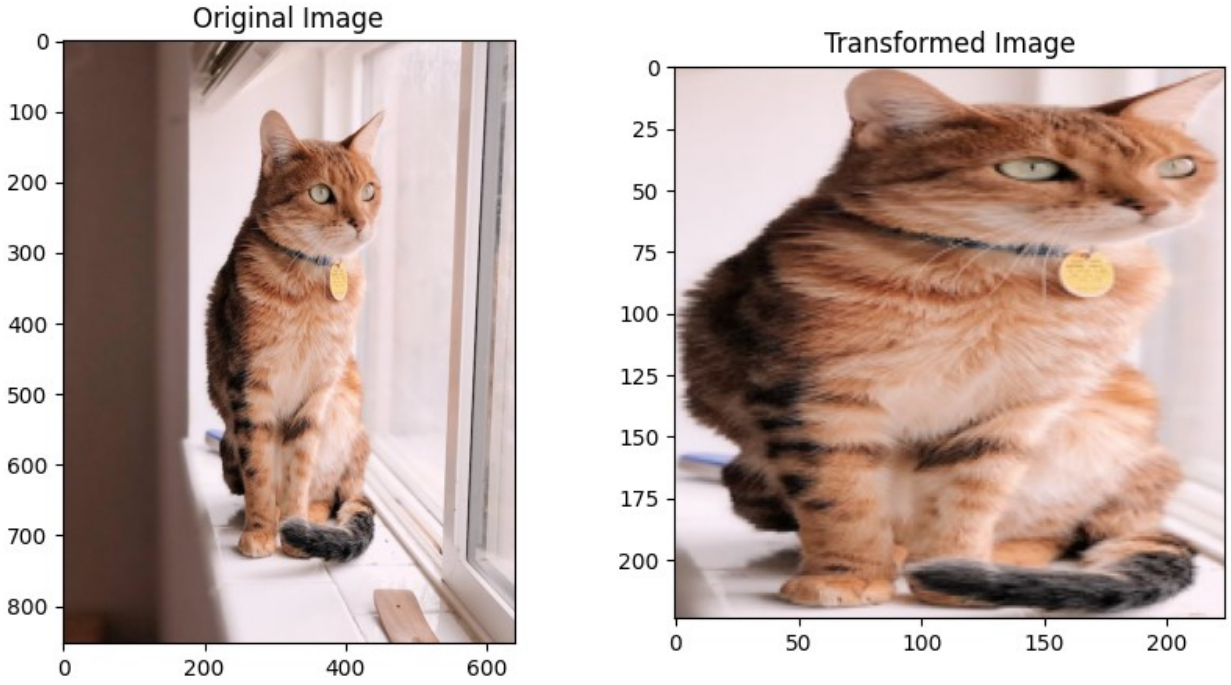
```
def flip(image, flip_code):         # 1 for horizontal flip, 0 for
vertical, -1 for both
  return cv2.flip(image, flip_code)


def rotate(image, angle, scale=1):
  h, w = image.shape[:2]
  center = (w // 2, h // 2)
  M = cv2.getRotationMatrix2D(center, angle, scale)
  return cv2.warpAffine(image, M, (w, h))


def resize(image, factorH, factorW):
  h, w = image.shape[:2]
  h = int(h * factorH)
  w = int(w * factorW)

  return cv2.resize(image, (w, h), interpolation=cv2.INTER_AREA)


def translate(image, shiftRightX, shiftDownY):    #shiftRightX and
shiftDownY are measured in pixels
  h, w = image.shape[:2]

  translation_matrix = np.float32([[1, 0, shiftRightX],
                                   [0, 1, shiftDownY]])

  return cv2.warpAffine(image, translation_matrix, (w, h))


def shear(image, shearX, shearY):
  h, w = image.shape[:2]

  shear_matrix = np.float32([[1, shearX, 0],
                             [shearY, 1, 0]])

  return cv2.warpAffine(image, shear_matrix, (w + int(shearX * h), h +
int(shearY * w)))


def stretch(image, stretchX, stretchY):
  h, w = image.shape[:2]

  stretch_matrix = np.float32([[stretchX, 0, 0],
                               [0, stretchY, 0]])

  return cv2.warpAffine(image, stretch_matrix, (int(w * stretchX),
int(h * stretchY)))
```

```
dataset1 = []
for i in range(1, 11):
  img = cv2.imread(f'/content/drive/MyDrive/Storage Extension/Colab
Notebooks/CSE463/Lab 1/23341134_UdoySaha_Lab1/Task2/Dataset1/{i}.jpg')
  dataset1.append(img)

def showImage(image):    # shows image in RGB format
  plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))

def showDataset(dataset, row):    # shows entire dataset
  plt.figure(figsize=(20, 12))
  for sbplt in range(row):
    for img in range(5):
      plt.subplot(2, 5, sbplt*5 + img + 1)
      showImage(dataset[sbplt*5 + img])
      plt.title(f'Image {sbplt*5 + img + 1}')

showDataset(dataset1, 2)
```

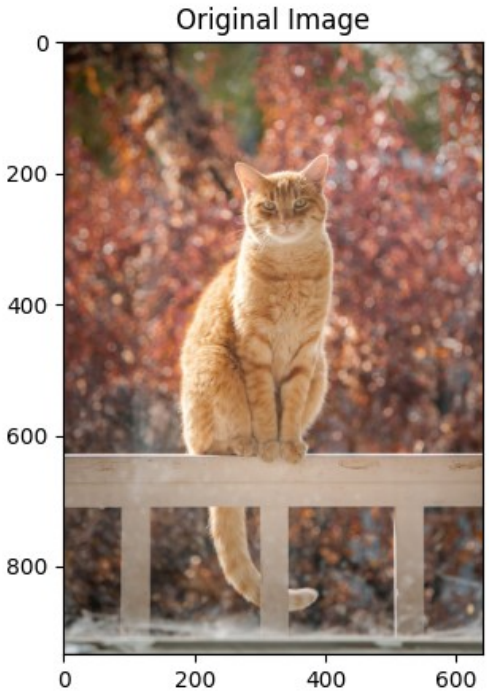


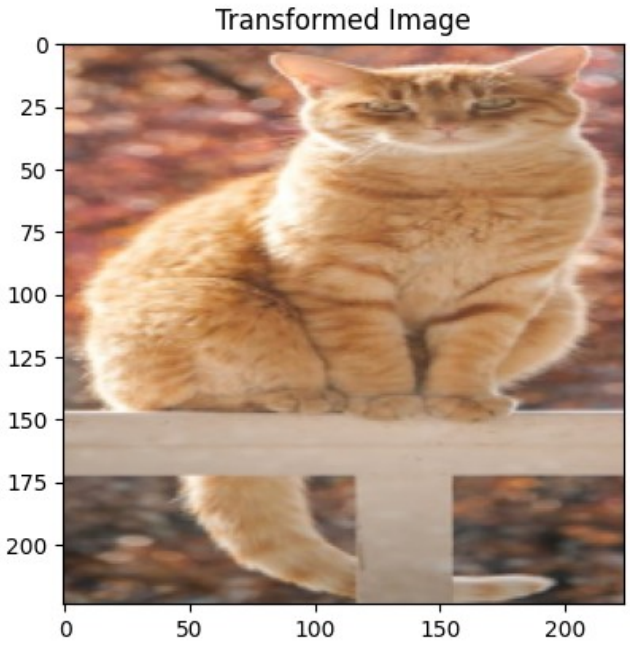

```
# Image 1 needs to be cropped so that the cat object is present only.
# Also, the image should be resized into a standard shape to make it
ready to apply various models.

image1 = dataset1[0]
image1 = crop(image1, 95, 200, 650, 255)
image1 = resize(image1, 224/image1.shape[0], 224/image1.shape[1])
```

```
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
showImage(dataset1[0])
plt.title('Original Image')
plt.subplot(1, 2, 2)
showImage(image1)
plt.title('Transformed Image')

dataset1[0] = image1
```

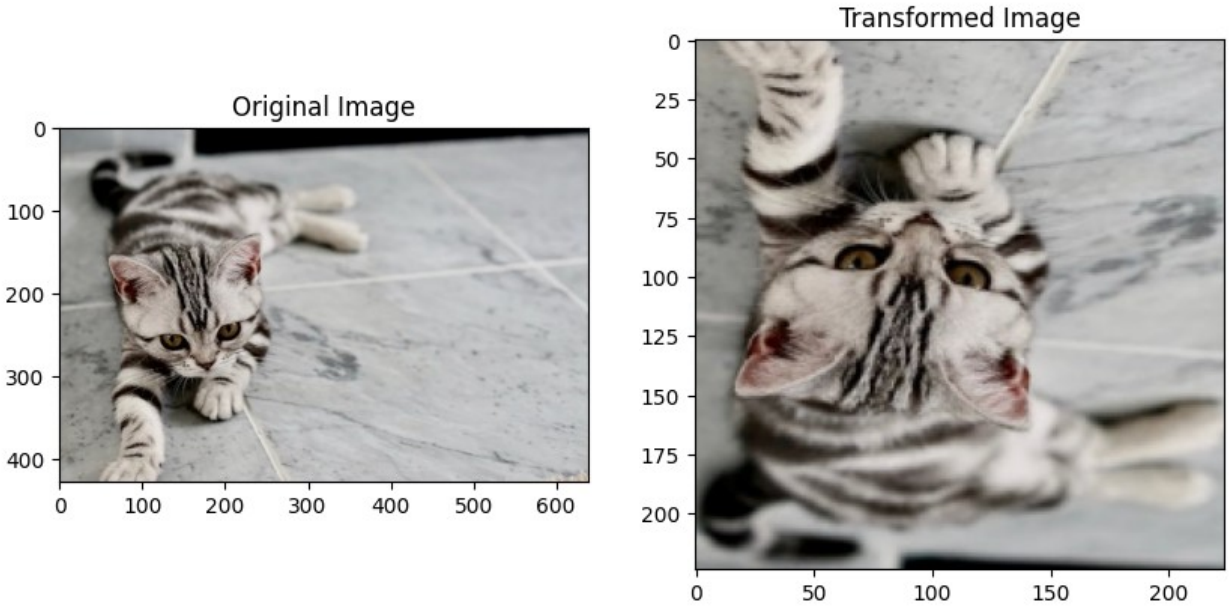



```
# Image 2 needs to be cropped so that the cat object is present only.
# Also, the image should be resized into a standard shape to make it
ready to apply various models.

image2 = dataset1[1]
image2 = crop(image2, 170, 170, 700, 250)
image2 = resize(image2, 224/image2.shape[0], 224/image2.shape[1])

plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
showImage(dataset1[1])
plt.title('Original Image')
plt.subplot(1, 2, 2)
showImage(image2)
plt.title('Transformed Image')

dataset1[1] = image2
```



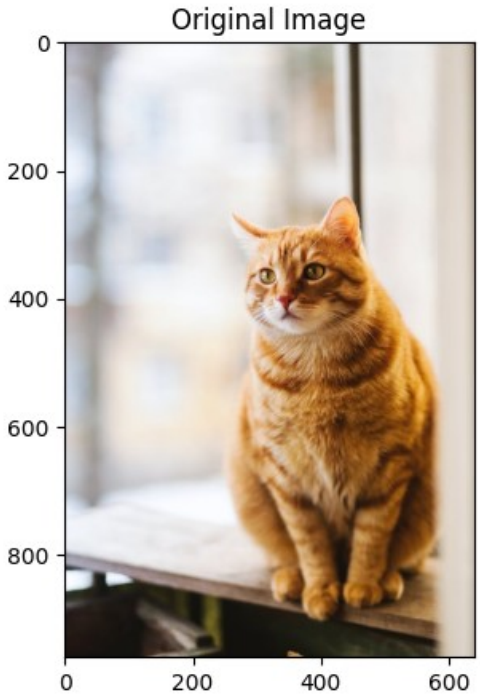


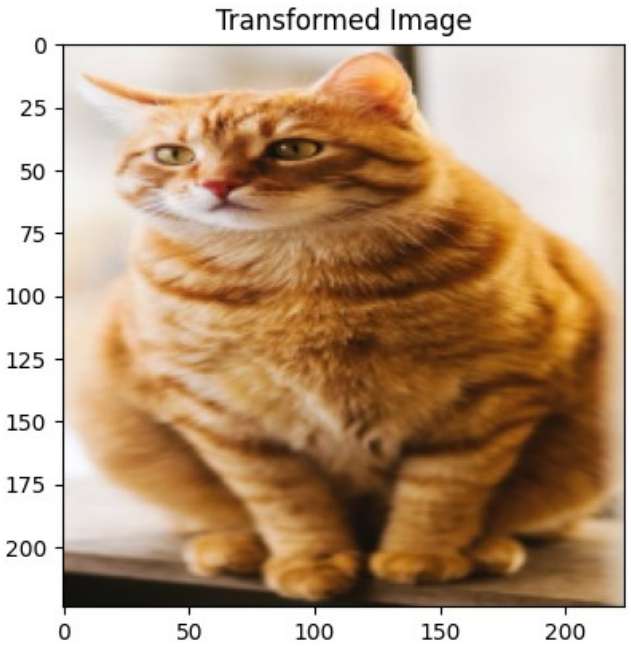

```
# Image 3 needs to be cropped so that the cat object is present only.
# Also, the image should be resized into a standard shape to make it
ready to apply various models.
# Additionally flipping the image vertically to add variation to the
dataset.

image3 = dataset1[2]
image3 = crop(image3, 25, 35, 400, 330)
image3 = resize(image3, 224/image3.shape[0], 224/image3.shape[1])
image3 = flip(image3, 0)

plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
showImage(dataset1[2])
plt.title('Original Image')
plt.subplot(1, 2, 2)
showImage(image3)
plt.title('Transformed Image')

dataset1[2] = image3
```



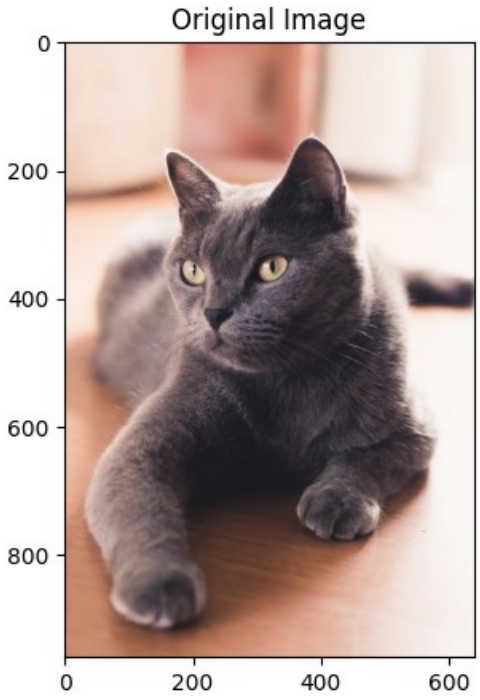

```
# Image 4 needs to be cropped so that the cat object is present only.
# Also, the image should be resized into a standard shape to make it
ready to apply various models.
# Additionally flipping the image vertically to add variation to the
dataset.

image4 = dataset1[3]
image4 = crop(image4, 25, 175, 375, 375)
image4 = rotate(image4, -90)
image4 = resize(image4, 224/image4.shape[0], 224/image4.shape[1])

plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
showImage(dataset1[3])
plt.title('Original Image')
plt.subplot(1, 2, 2)
showImage(image4)
plt.title('Transformed Image')

dataset1[3] = image4
```

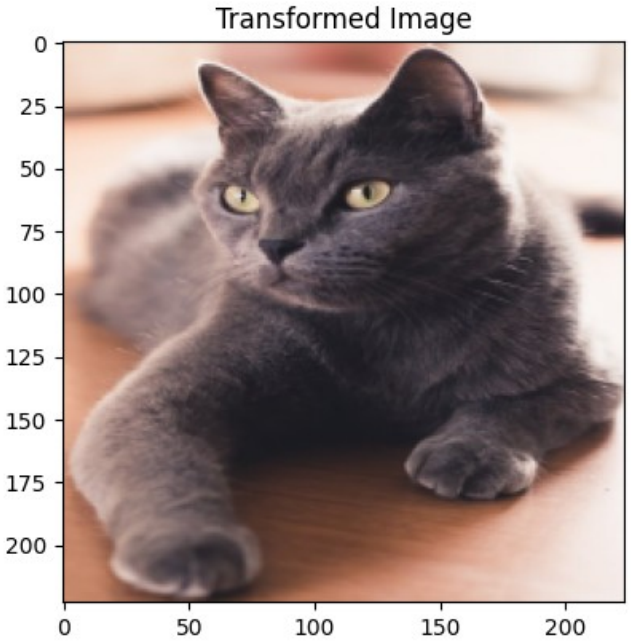

```
# Image 5 needs to be cropped so that the cat object is present only.
# Also, the image should be resized into a standard shape to make it
ready to apply various models.

image5 = dataset1[4]
image5 = crop(image5, 230, 250, 680, 340)
image5 = resize(image5, 224/image5.shape[0], 224/image5.shape[1])

plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
showImage(dataset1[4])
plt.title('Original Image')
plt.subplot(1, 2, 2)
showImage(image5)
plt.title('Transformed Image')

dataset1[4] = image5
```



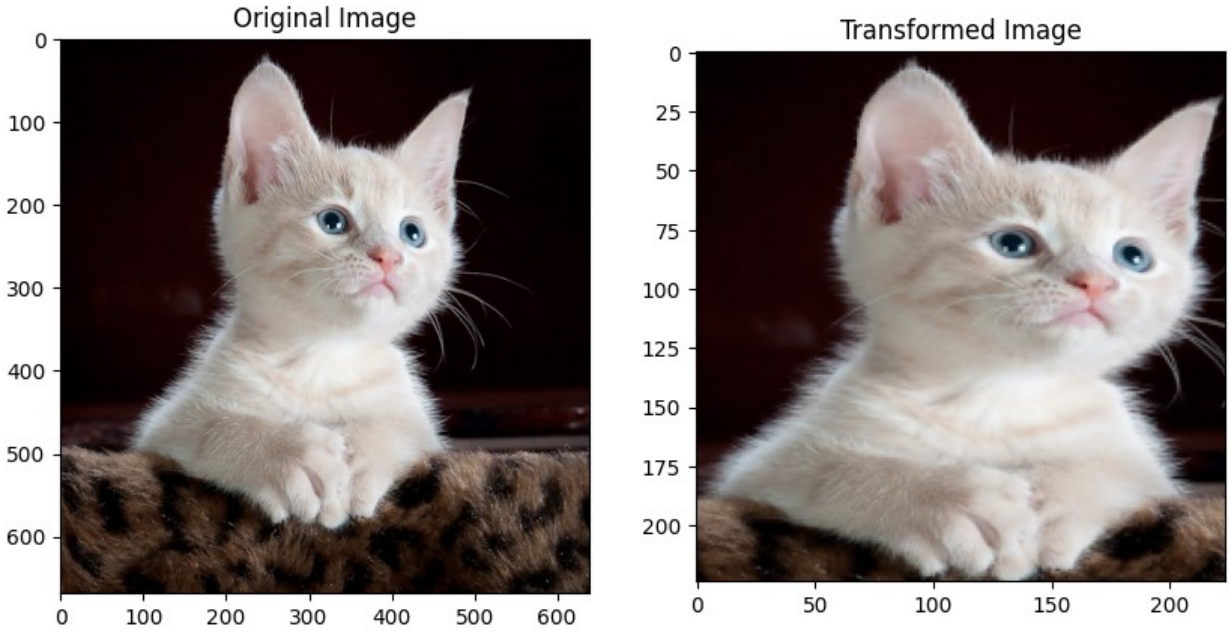



```
# Image 6 needs to be cropped so that the cat object is present only.
# Also, the image should be resized into a standard shape to make it
ready to apply various models.

image6 = dataset1[5]
image6 = crop(image6, 140, 25, 780, 555)
image6 = resize(image6, 224/image6.shape[0], 224/image6.shape[1])

plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
showImage(dataset1[5])
plt.title('Original Image')
plt.subplot(1, 2, 2)
showImage(image6)
plt.title('Transformed Image')

dataset1[5] = image6
```



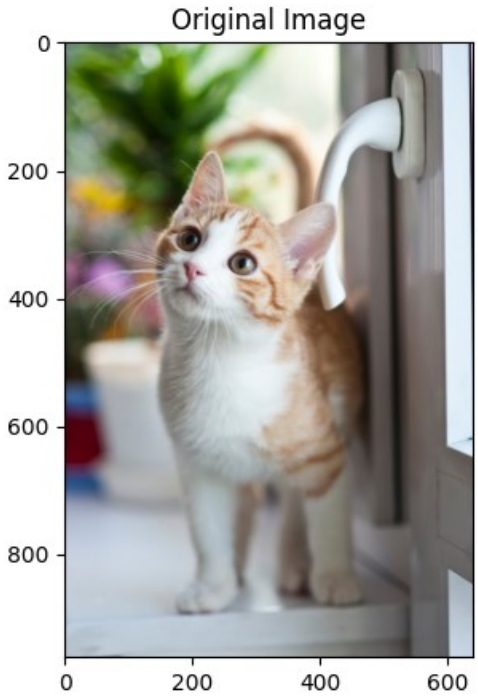


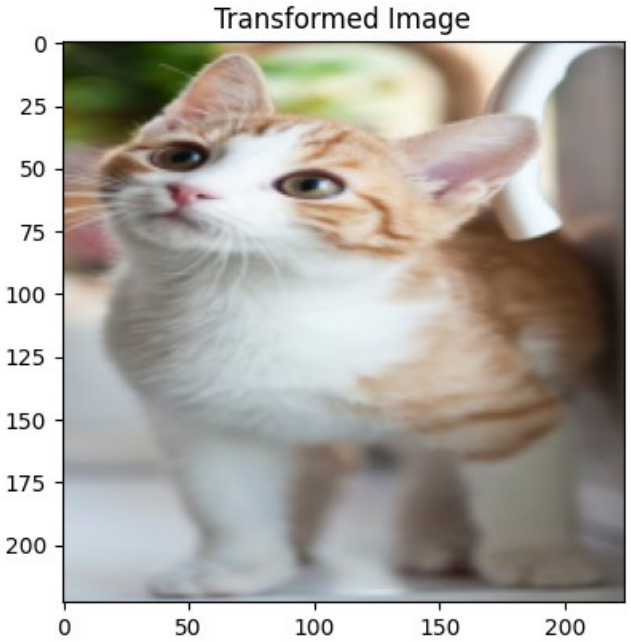

```
# Image 7 needs to be cropped so that the cat object is present only.
# Also, the image should be resized into a standard shape to make it
ready to apply various models.

image7 = dataset1[6]
image7 = crop(image7, 10, 75, 580, 425)
image7 = resize(image7, 224/image7.shape[0], 224/image7.shape[1])

plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
showImage(dataset1[6])
plt.title('Original Image')
plt.subplot(1, 2, 2)
showImage(image7)
plt.title('Transformed Image')

dataset1[6] = image7
```



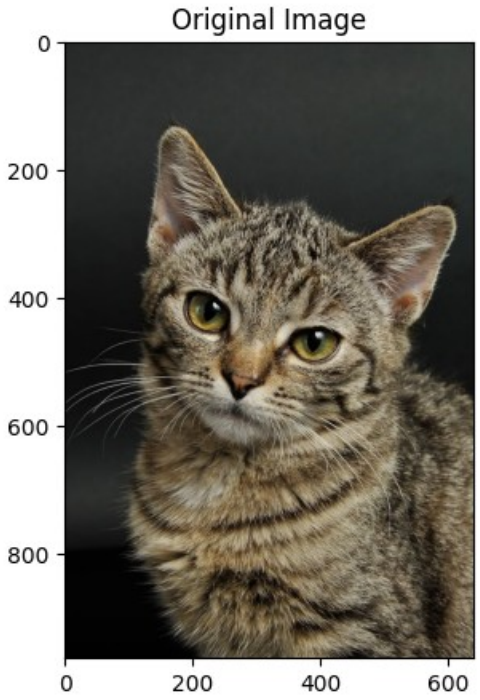

```
# Image 8 needs to be cropped so that the cat object is present only.
# Also, the image should be resized into a standard shape to make it
ready to apply various models.

image8 = dataset1[7]
image8 = crop(image8, 75, 20, 350, 250)
image8 = resize(image8, 224/image8.shape[0], 224/image8.shape[1])

plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
showImage(dataset1[7])
plt.title('Original Image')
plt.subplot(1, 2, 2)
showImage(image8)
plt.title('Transformed Image')

dataset1[7] = image8
```

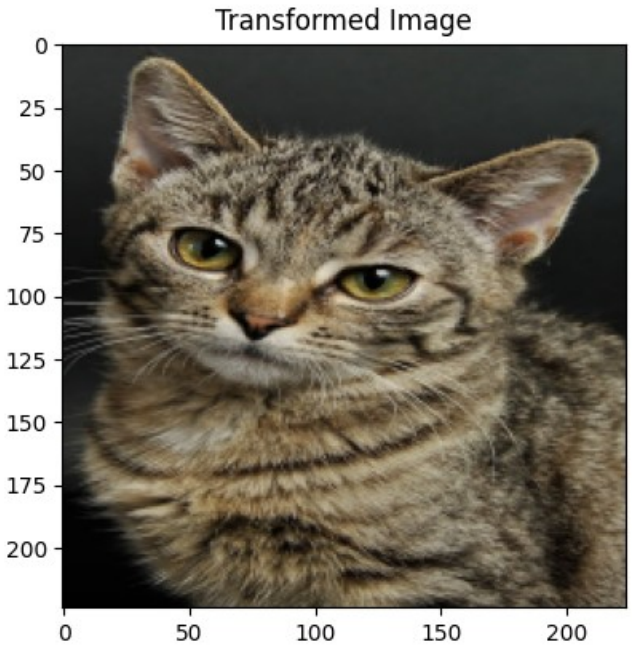

```
# Image 9 needs to be cropped so that the cat object is present only.
# Also, the image should be resized into a standard shape to make it
ready to apply various models.

image9 = dataset1[8]
image9 = crop(image9, 155, 120, 745, 360)
image9 = resize(image9, 224/image9.shape[0], 224/image9.shape[1])

plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
showImage(dataset1[8])
plt.title('Original Image')
plt.subplot(1, 2, 2)
showImage(image9)
plt.title('Transformed Image')

dataset1[8] = image9
```






```
# Image 10 needs to be cropped so that the cat object is present only.
# Also, the image should be resized into a standard shape to make it
ready to apply various models.

image10 = dataset1[9]
image10 = crop(image10, 110, 80, 945, 800)
image10 = resize(image10, 224/image10.shape[0], 224/image10.shape[1])

plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
showImage(dataset1[9])
plt.title('Original Image')
plt.subplot(1, 2, 2)
showImage(image10)
plt.title('Transformed Image')

dataset1[9] = image10
```



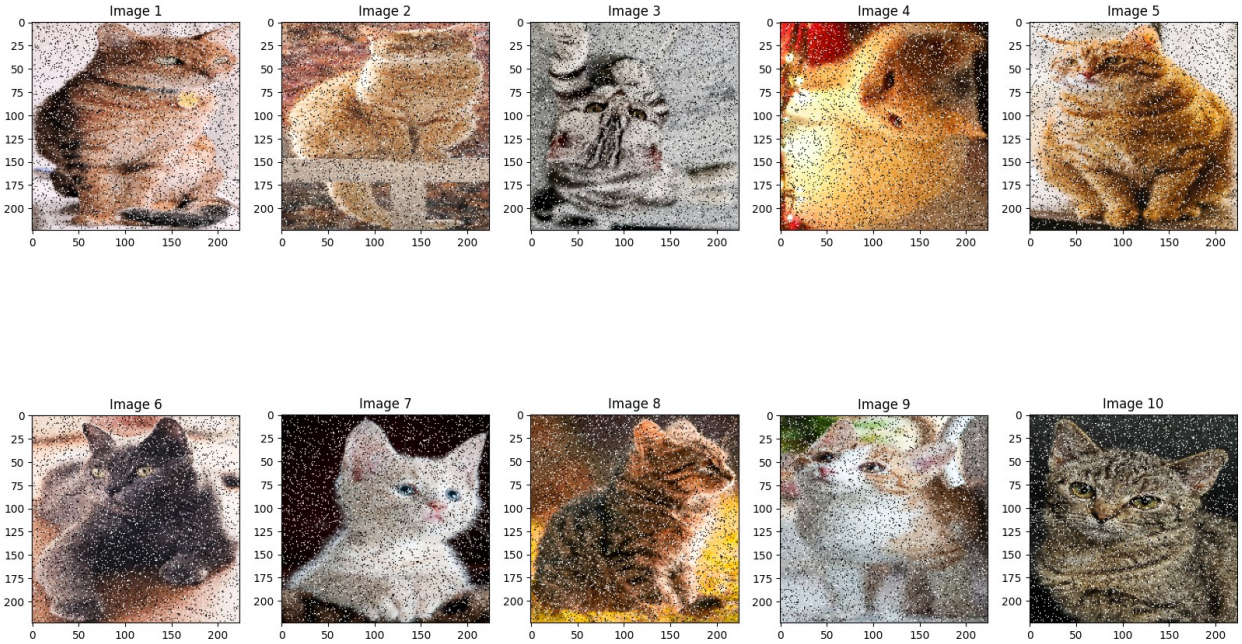



```
showDataset(dataset1, 2)
```

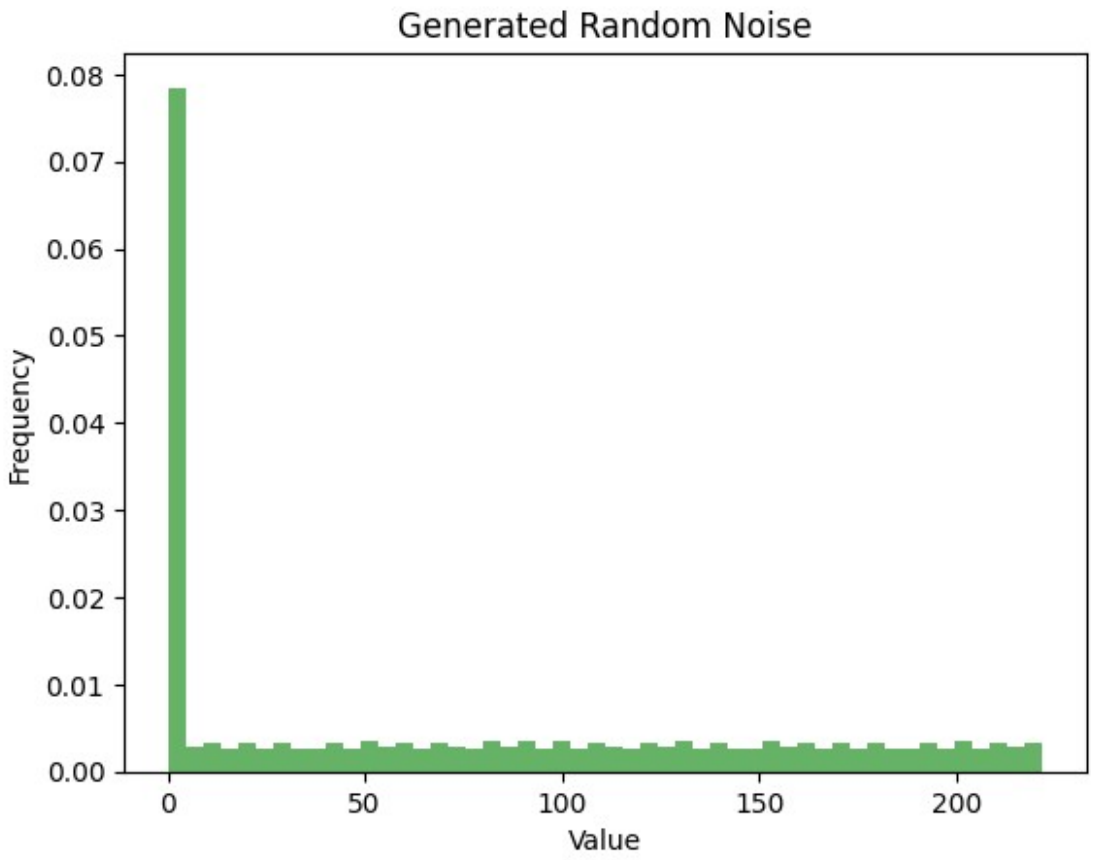

```
noise_dataset = []
noisy_image_dataset = []

for image in dataset1:
  prob = 0.05  # Probability of noise
```

```
  noisy_image = np.copy(image)

  # Number of 'salt' pixels
  num_salt = np.ceil(prob * image.size * 0.5).astype(int)
  # Number of 'pepper' pixels
  num_pepper = np.ceil(prob * image.size * 0.5).astype(int)

  # Apply 'salt' noise (white pixels)
  coords_salt = [np.random.randint(0, i - 1, num_salt) for i in
image.shape]
  noisy_image[coords_salt[0], coords_salt[1]] = 255

  # Apply 'pepper' noise (black pixels)
  coords_pepper = [np.random.randint(0, i - 1, num_pepper) for i in
image.shape]
  noisy_image[coords_pepper[0], coords_pepper[1]] = 0


  noise_dataset.extend(np.array(coords_salt).flatten())
  noise_dataset.extend(np.array(coords_pepper).flatten())
  noisy_image_dataset.append(noisy_image)

# Before adding noise
showDataset(dataset1, 2)
```


```
# After adding noise
showDataset(noisy_image_dataset, 2)
```

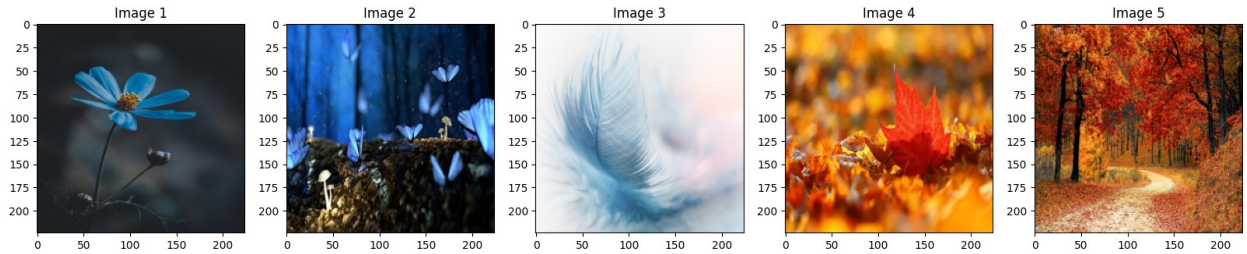



```
# Plotting overall noise dataset
plt.hist(noise_dataset, bins=50, density=True, alpha=0.6, color='g')
plt.title('Generated Random Noise')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```



Task 3

```
dataset2 = []
for i in range(1, 6):
  img = cv2.imread(f'/content/drive/MyDrive/Storage Extension/Colab Notebooks/CSE463/Lab 1/23341134_UdoySaha_Lab1/Task3/Dataset2/{i}.jpg')
  img = resize(img, 224/img.shape[0], 224/img.shape[1])
  dataset2.append(img)

showDataset(dataset2, 1)
```




```
# Blending the images together
blended_image = dataset2[0]*0.2 + dataset2[1]*0.3 + dataset2[2]*0.1 +
```

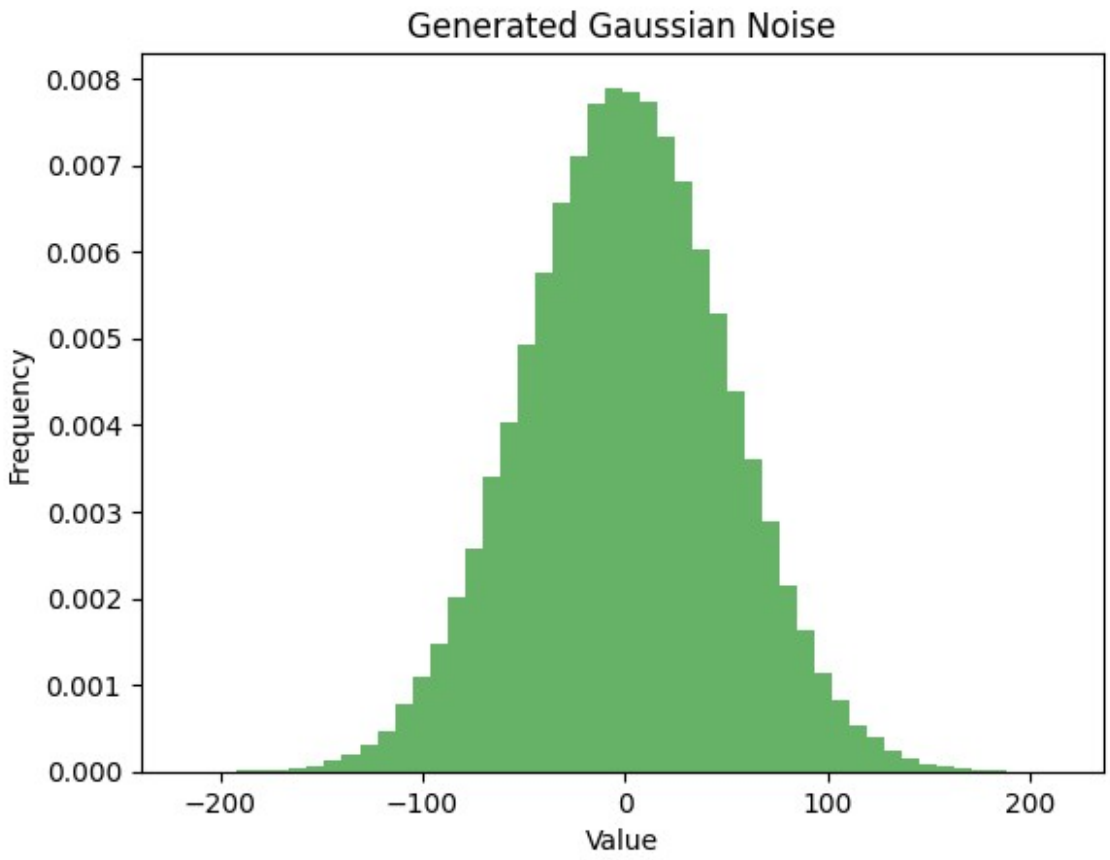
```
dataset2[3]*0.1 + dataset2[4]*0.6
blended_image = np.clip(blended_image, 0, 255).astype('uint8')
showImage(blended_image)
plt.axis('off')
```

```
(-0.5, 223.5, 223.5, -0.5)
```

```
noise_values = np.random.randn(*img.shape) * 50
vintage_image = blended_image + noise_values
vintage_image = np.clip(vintage_image, 0, 255).astype('uint8')
showImage(vintage_image)
plt.axis('off')
```

```
(-0.5, 223.5, 223.5, -0.5)
```

```
flattened_noise = noise_values.flatten()

plt.hist(flattened_noise, bins=50, density=True, alpha=0.6, color='g')
plt.title('Generated Gaussian Noise')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```


Generated Gaussian Noise
Frequency
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
−200
−100
0
100
200
Value