2. Implemente função que calcula a entropia (descritor de textura em imagens, calculada utilizando o histograma) de uma imagem.

    1. Converta a imagem em tons de cinza, se ainda não estiver
    2. Calcular o histograma da imagem usando numpy
    3. Calcule a probabilidade de ocorrência de cada nível de intensidade na imagem
        a. Calcular histograma
        b. Normalização do histograma
    4. Filtrar probabilidades não nulas para evitar problemas com log2(0)
    5. Calcular a entropia da imagem (Shannon: Entropia = -Σ(p * log2(p)))
    6. Some os resultados de todos os níveis de intensidade para obter a entropia total da imagem.

```
def image_entropy(img):
    # Calcular o histograma da imagem usando numpy
    hist, _ = np.histogram(img, bins=256, range=[0, 256])

    # Normalizar o histograma para obter as probabilidades de ocorrência
    probabilities = hist / float(img.size)

    # Filtrar probabilidades não nulas para evitar problemas com log2(0)
    non_zero_probabilities = probabilities[probabilities > 0]

    # Calcular a entropia da imagem
    entropy = -np.sum(non_zero_probabilities *
np.log2(non_zero_probabilities))

    return entropy
```

3. Proponha uma representação para imagem que permita que a transformada de Fourier seja utilizada para obtenção de medidas invariantes à rotação.

Representação com descritores de Hu.

Pelos coeficientes da transformada podemos expressar as informações globais da curvatura de um objeto.

Os momentos invariantes (descritores) são 7 momentos de Hu e ele são relativamente invariantes à translação, rotação e escala.
***Não** são diretamente obtidos por meio da Transformada de Fourier, mas sim por cálculos específicos baseados em momentos.

P+Q determina a ordem desse momento para diversos tipos de predições, um momento.
e.g:

- P(0)Q(0) determina a área da região, o centro dela.

- P(0)Q(1) define a projeção horizontal.
- P(1)Q(0) define a projeção vertical.
- Se você pegar o P(0)Q(1) e dividir por P(0)Q(0) irá definir a posição centróide do objeto.)

```
coord = measure.find_contours(bat_bin, fully_connected='low')
ncoord = np.empty(shape=[0, 2])
for contour in coord:
ncoord = np.append(ncoord, contour, axis=0)


z = frdescp(ncoord)
nz = ifrdescp(z, 20)


lin, col = bat.shape[:2]
nimg = bound2img(nz, lin, col)
nimg2 = bound2img(ncoord, lin, col)


fig, ax = plt.subplots(1,2,figsize=(10,10))
ax[0].imshow(nimg2, cmap=plt.cm.gray)
ax[1].imshow(nimg, cmap='gray')
```

4. Explique pra que é usado o parâmetro de regularização

O parâmetro de regularização é usado para **evitar o overfitting** em aprendizado de padrões de imagens.

Ele a**diciona uma penalidade à função de perda durante o treinamento**, tornando o modelo mais simples e **reduzindo a influência de parâmetros individuais**. Isso ajuda a obter um modelo com melhor capacidade de generalização e representações mais significativas das imagens.

5. Explique a diferença entre o *dropout regular* e o *dropout espacial.*

O dropout **regular desativa neurônios individuais**, enquanto o dropout **espacial desativa mapas de recursos inteiros**.

A escolha entre eles depende do tipo de arquitetura de rede neural e da natureza da tarefa em questão. **Ambos** têm o objetivo de **regularizar o modelo** e melhorar a sua capacidade de generalização.

6. Implemente função que multiplica a imagem de entrada pelo fator (-1)^(x+y).

```
def my_fftshift(image):
    height, width = image.shape
```

```
    factor = np.fromfunction(lambda x, y: (-1) ** (x + y), (height, width),
dtype=int)

    new_image = image * factor

return new_image
```