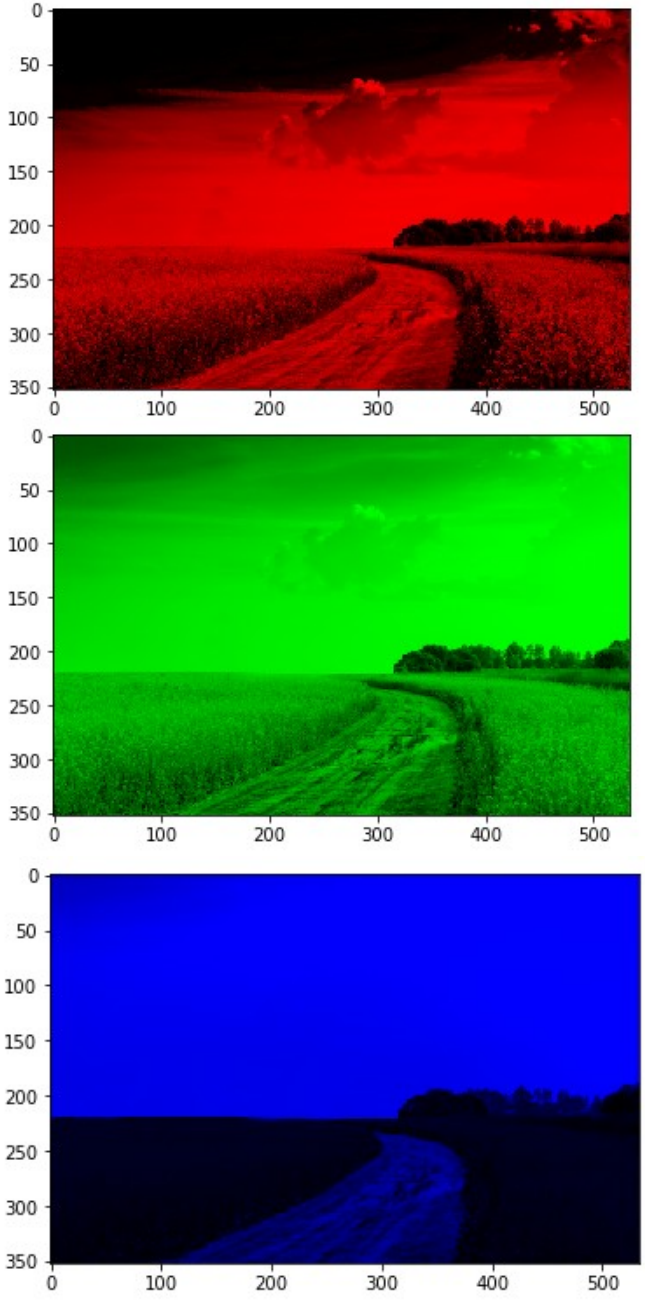
Tarefa 1

- Por enquanto, estamos usando duas bibliotecas:
  1)import matplotlib.pyplot as plt
  2)import numpy as np

- E usamos as funções de leitura “imread” e apresentação “imshow” do matplotlib
  1) f = plt.imread('field.png')
  2) plt.imshow(f)

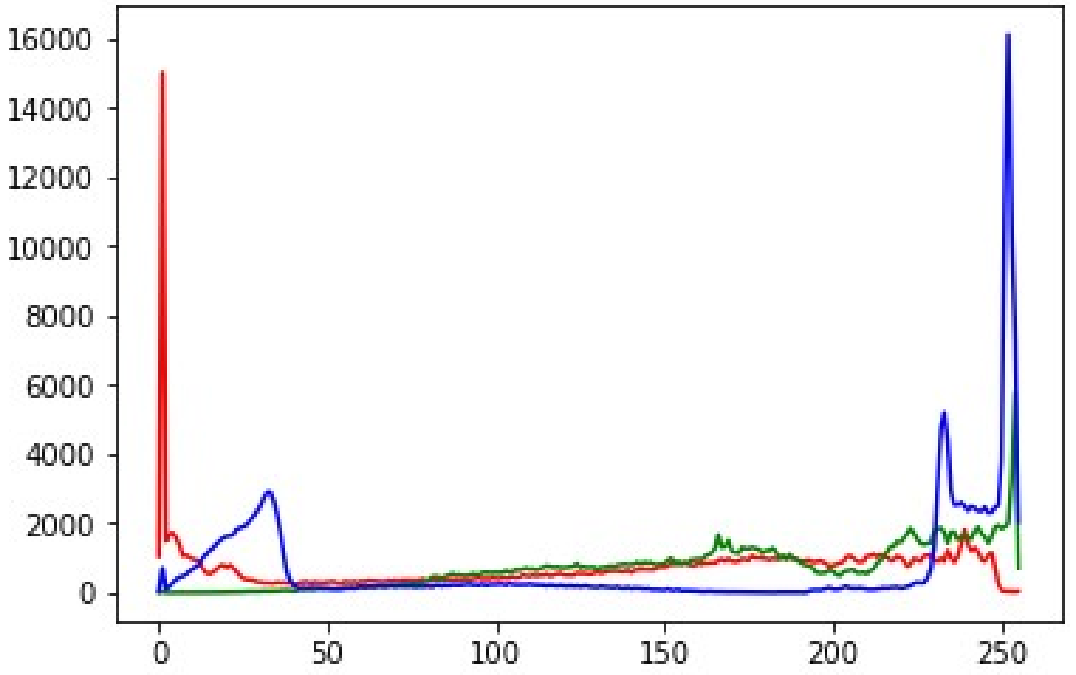
Formato RGB

- f[:,:,0] → Camada R (red)
  - r = f.copy()
  - r[:,:,1] = 0
  - r[:,:,2] = 0
  - plt.imshow(r)
- f[:,:,1] → Camada G (green)
-
-
- f[:,:,2] → Camada B (blue)



Histograma

- hr,bins = np.histogram(f[:,:,0], bins=256)
- hg,bins = np.histogram(f[:,:,1], bins=256)
- hb,bins = np.histogram(f[:,:,2], bins=256)
- plt.plot(hr, color='r')
- plt.plot(hg, color='g')
- plt.plot(hb, color='b')



hr: array([1052, 15024, 1494, ...,   31,   28,   43], dtype=int64)

Apenas com estes conceitos, o que é possível se fazer?

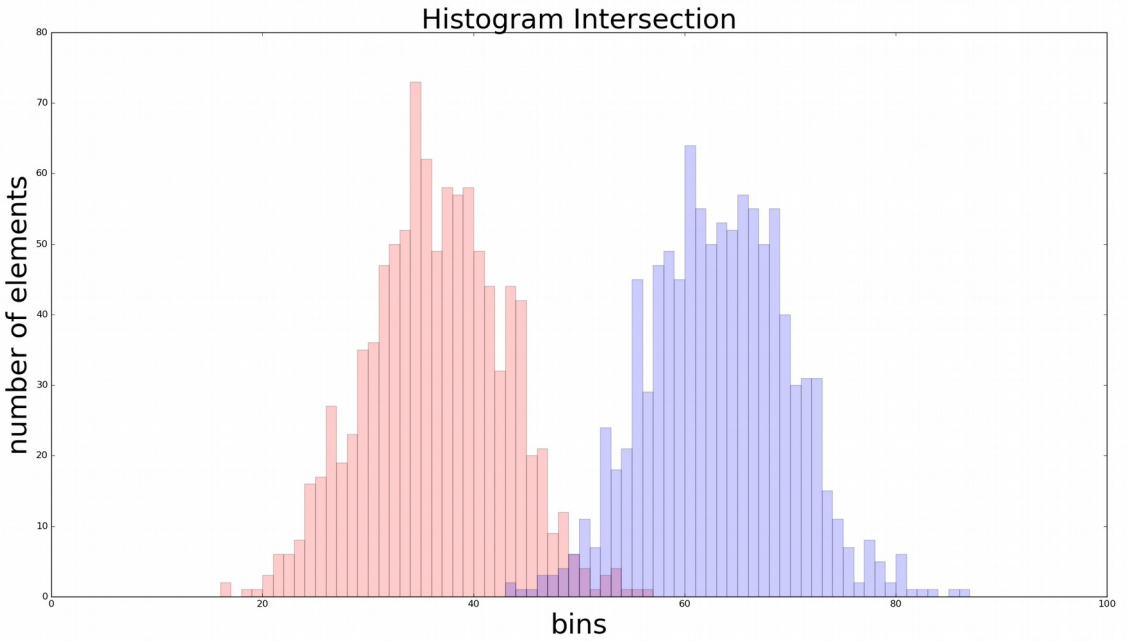
É possível comparar imagens

Como?

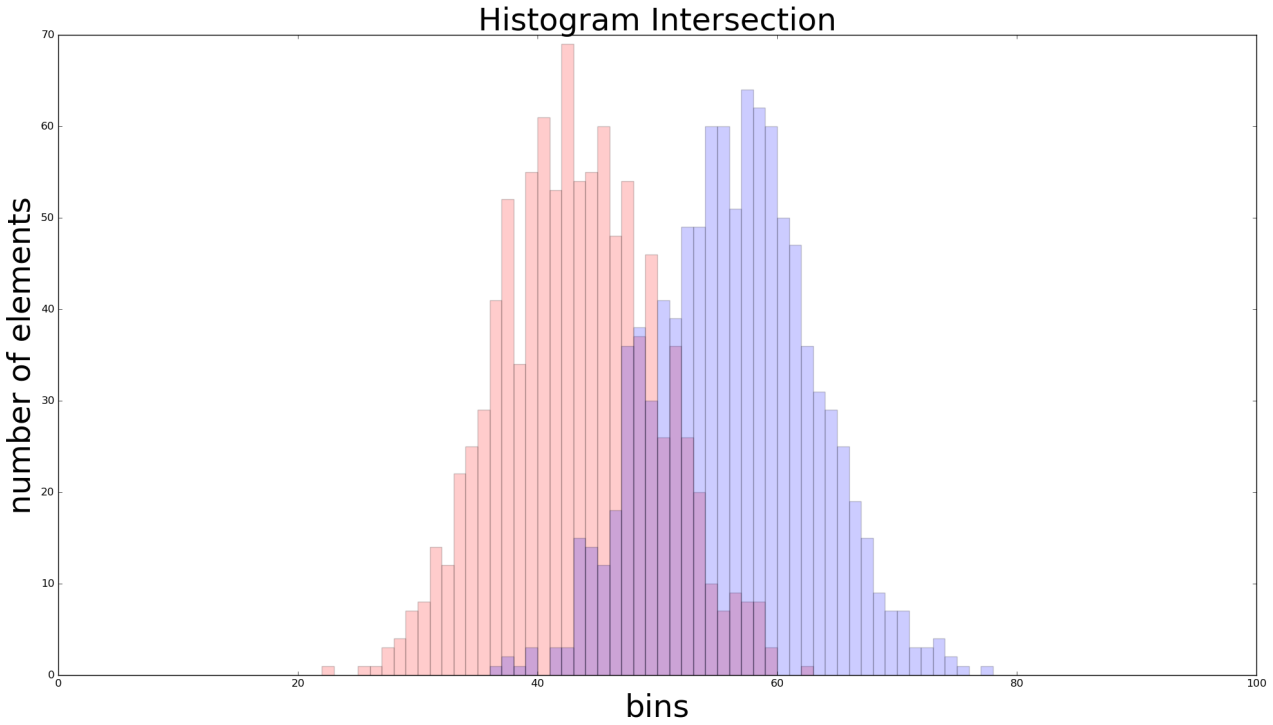
- Comparando histogramas!
- Digamos que temos duas imagens em escala de cinza, uma com histograma *I* e outra com histograma *M*, cada uma com *n* bins

$$\sum_{j=1}^{n} min(I_j, M_j)$$

$$\frac{\sum_{j=1}^{n} min(I_j, M_j)}{\sum_{j=1}^{n} M_j}$$

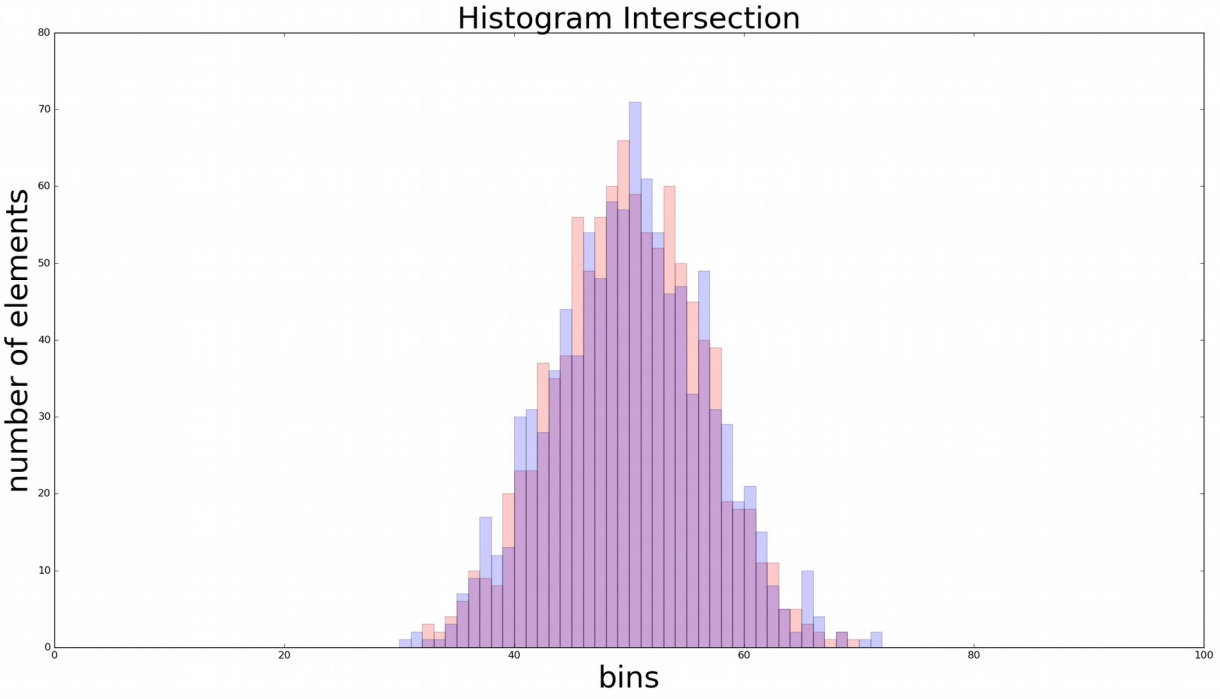


0.035


Histogram Intersection
number of elements
bins
0
10
20
30
40
50
60
70
0
20
40
60
80
100

0.329

0.89

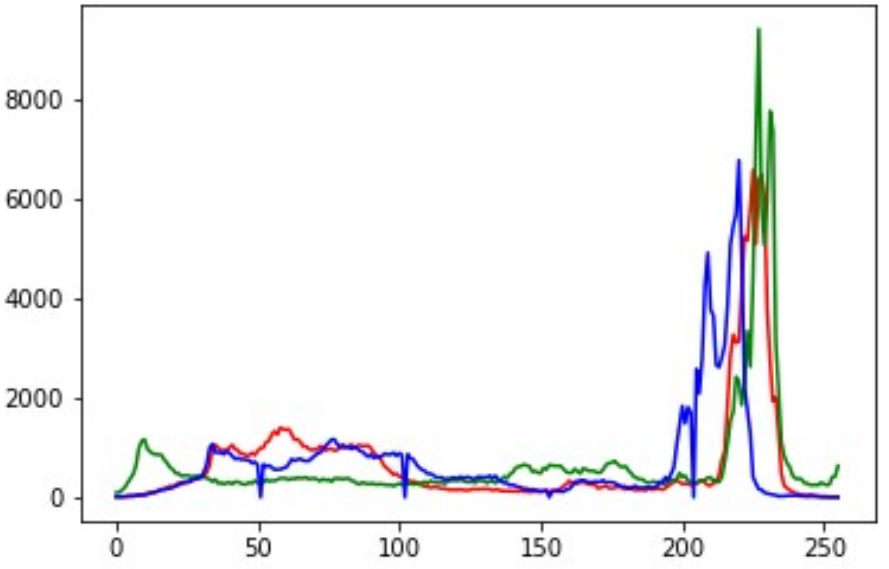

Histogram Intersection
number of elements
bins
0
10
20
30
40
50
60
70
80
0
20
40
60
80
100

Código que faz a comparação

- def return_intersection(hist_1, hist_2):
- minima = np.minimum(hist_1, hist_2)
- intersection = np.true_divide(np.sum(minima), np.sum(hist_2))
- return intersection

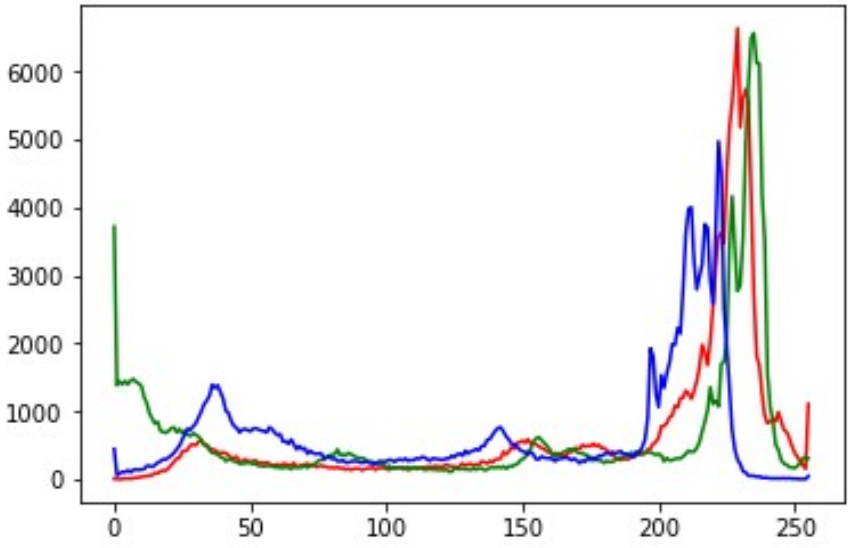
Um primeiro exemplo: apenas dois modelos

- image = plt.imread('hulk.png')
- hr,bins = np.histogram(image[:,:,0], bins=256)
- hg,bins = np.histogram(image[:,:,1], bins=256)
- hb,bins = np.histogram(image[:,:,2], bins=256)
- plt.plot(hr, color='r')
- plt.plot(hg, color='g')
- plt.plot(hb, color='b')



Um primeiro exemplo: apenas dois modelos

- image2 = plt.imread('maravilha.png')
- hr2,bins = np.histogram(image2[:,:,0], bins=256)
- hg2,bins = np.histogram(image2[:,:,1], bins=256)
- hb2,bins = np.histogram(image2[:,:,2], bins=256)
- plt.plot(hr2, color='r')
- plt.plot(hg2, color='g')
- plt.plot(hb2, color='b')



Comparando

- r = return_intersection(hr, hr)
- g = return_intersection(hg, hg)
- b = return_intersection(hb, hb)
- compatibilidade = r + g + b
-
- O resultado do histograma com ele mesmo, resulta no máximo, que é 3,0

Comparando

- r = return_intersection(hr, hr2)
- g = return_intersection(hg, hg2)
- b = return_intersection(hb, hb2)
- compatibilidade = r + g + b
-
- O resultado do histograma com ele mesmo, resulta no máximo, que é 2,1

1ª parte da Tarefa

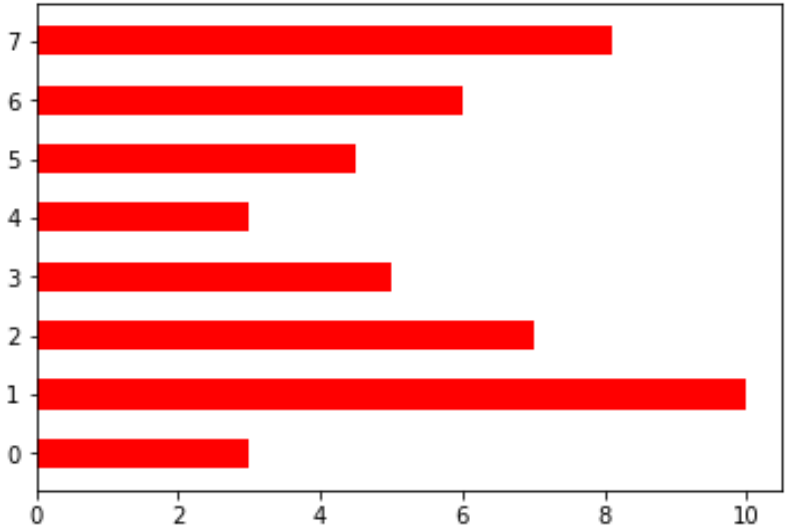
- Sua base de dados tem 8 imagens-modelo:
  1) America
  2) Batman
  3) Ferro
  4) Flash
  5) Hulk
  6) Maravilha
  7) Super
  8) Wolverine

1ª parte da Tarefa

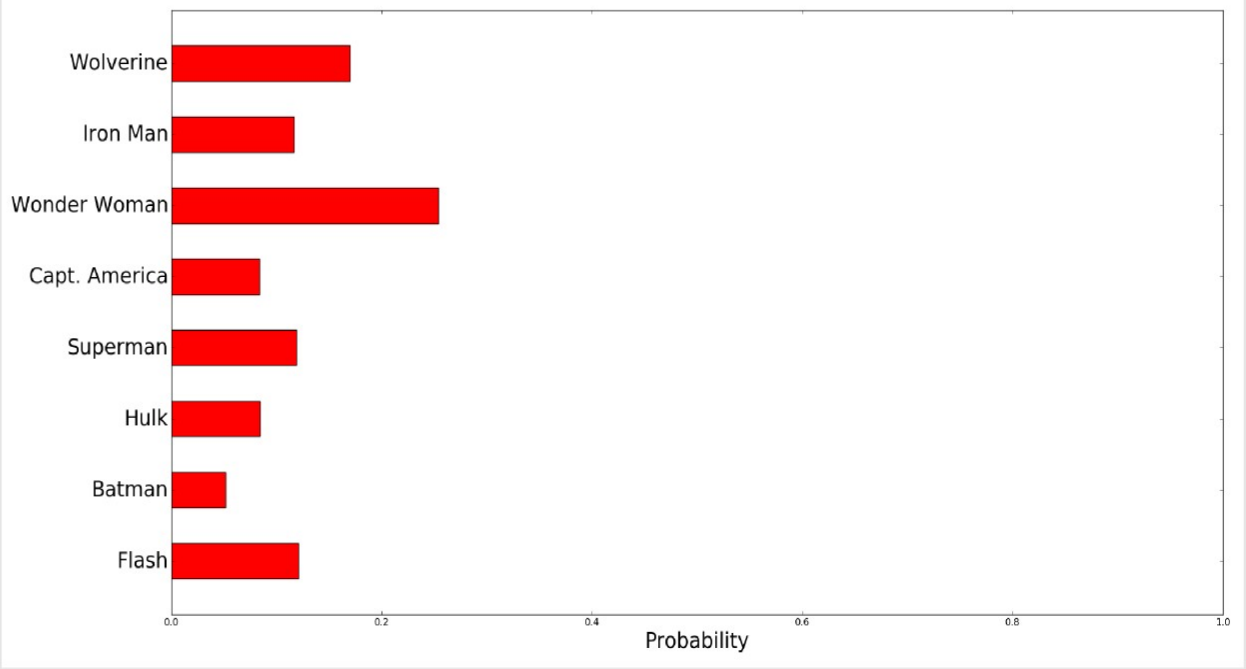
- E tem 5 imagens-teste:
  1) QUEM1
  2) QUEM2
  3) QUEM3
  4) QUEM4
  5) QUEM5

2ª parte da Tarefa

- Armazene o resultado em um array, e depois apresente-os em um gráfico de barras. Veja um exemplo
- y = [3, 10, 7, 5, 3, 4.5, 6, 8.1]
- N = len(y)
- x = range(N)
- width = 0.5
- plt.barh(x, y, width, color="red")
- plt.show()



Com o gráfico é possível avaliar quem é o mais parecido



3ª parte da Tarefa

- Mantenha as 8 imagens-modelo e faça os testes usando a base 2 como testes.
- Use outras formas de comparação de histograma, que são descritas em: http://www.pyimagesearch.com/2014/07/14/3-ways-compare-histograms-using-opencv-python/
-
- Com as notas formas de comparação de histogramas, refaça os testes com todas as imagens iniciadas com “quem”.

4ª parte da Tarefa

- Use o código resultante da 3ª parte da tarefa e use os mesmos procedimentos para Base3.zip

- Para a tarefa, faça o upload de todos os seus códigos e um relatório explicando o que você fez e quais foram os resultados obtidos.

- Um dos alunos apresentará o trabalho e o resultado em sala de aula.