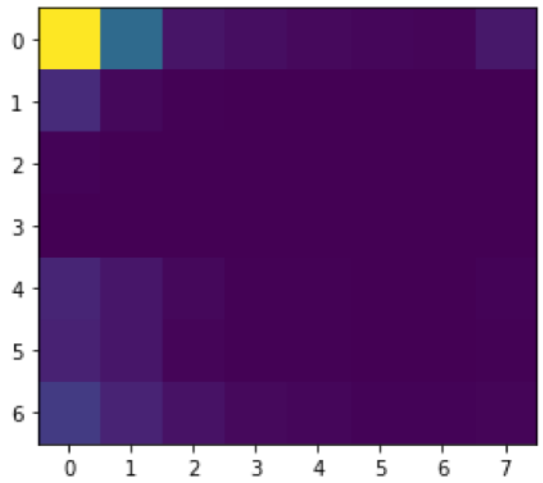
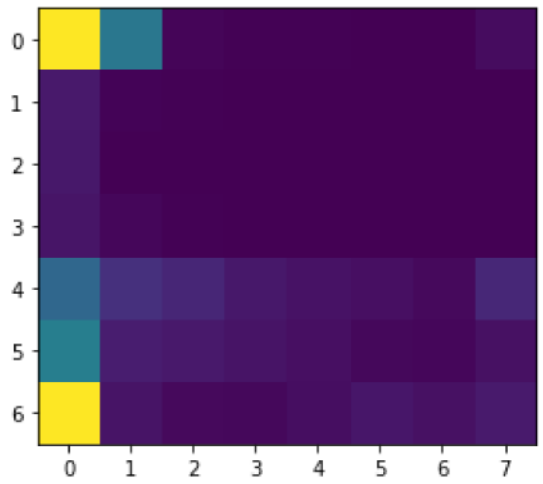
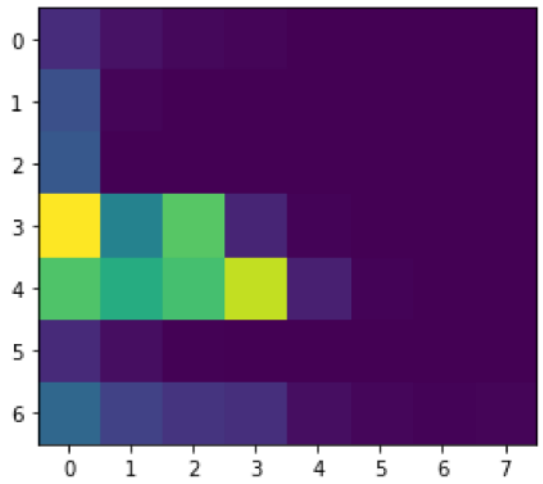
```
import cv2
import math
import sys
import time
import glob
import numpy as np
import seaborn as sns
import cv2 as cv
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score


#Funciones para realizar la interpolacion

# Kernel de interpolacion
def u(s, a):
    if (abs(s) >= 0) & (abs(s) <= 1):
        return (a+2)*(abs(s)**3)-(a+3)*(abs(s)**2)+1
    elif (abs(s) > 1) & (abs(s) <= 2):
        return a*(abs(s)**3)-(5*a)*(abs(s)**2)+(8*a)*abs(s)-4*a
    return 0


# Padding
def padding(img, H, W, C):
    zimg = np.zeros((H+4, W+4, C))
    zimg[2:H+2, 2:W+2, :C] = img

    # Rellenar la primera/ultima fila y columna
    zimg[2:H+2, 0:2, :C] = img[:, 0:1, :C]
    zimg[H+2:H+4, 2:W+2, :] = img[H-1:H, :, :]
    zimg[2:H+2, W+2:W+4, :] = img[:, W-1:W, :]
    zimg[0:2, 2:W+2, :C] = img[0:1, :, :C]

    # Rellenar los puntos perdidos
    zimg[0:2, 0:2, :C] = img[0, 0, :C]
    zimg[H+2:H+4, 0:2, :C] = img[H-1, 0, :C]
    zimg[H+2:H+4, W+2:W+4, :C] = img[H-1, W-1, :C]
    zimg[0:2, W+2:W+4, :C] = img[0, W-1, :C]
    return zimg


# Operacion Bicubica
def bicubic(img, ratio, a):

    # Tamanño de la imagen
    H, W, C = img.shape

    # Here H = Height, W = weight,
    # C = Numero de canales si la imagen esta a color
    img = padding(img, H, W, C)

    # Crer nueva imagen
    dH = math.floor(H*ratio)
    dW = math.floor(W*ratio)

    # Convertir en matriz
    dst = np.zeros((dH, dW, 3))

    h = 1/ratio

    print('Start bicubic interpolation')
    print('It will take a little while...')
    inc = 0

    for c in range(C):
        for j in range(dH):
            for i in range(dW):

                # Obtener las coordinadas de los valores cercanos
                x, y = i * h + 2, j * h + 2

                x1 = 1 + x - math.floor(x)
                x2 = x - math.floor(x)
                x3 = math.floor(x) + 1 - x
```

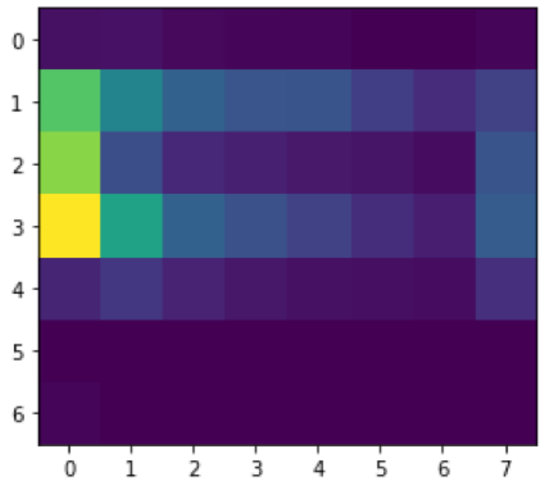
```
                x3 = math.floor(x) + 1 - x
                x4 = math.floor(x) + 2 - x

                y1 = 1 + y - math.floor(y)
                y2 = y - math.floor(y)
                y3 = math.floor(y) + 1 - y
                y4 = math.floor(y) + 2 - y

                # Considerando todos los 16 valores cercanos
                mat_l = np.matrix([[u(x1, a), u(x2, a), u(x3, a), u(x4, a)]])
                mat_m = np.matrix([[img[int(y-y1), int(x-x1), c],
                                    img[int(y-y2), int(x-x1), c],
                                    img[int(y+y3), int(x-x1), c],
                                    img[int(y+y4), int(x-x1), c]],
                                   [img[int(y-y1), int(x-x2), c],
                                    img[int(y-y2), int(x-x2), c],
                                    img[int(y+y3), int(x-x2), c],
                                    img[int(y+y4), int(x-x2), c]],
                                   [img[int(y-y1), int(x+x3), c],
                                    img[int(y-y2), int(x+x3), c],
                                    img[int(y+y3), int(x+x3), c],
                                    img[int(y+y4), int(x+x3), c]],
                                   [img[int(y-y1), int(x+x4), c],
                                    img[int(y-y2), int(x+x4), c],
                                    img[int(y+y3), int(x+x4), c],
                                    img[int(y+y4), int(x+x4), c]]])
                mat_r = np.matrix(
                    [[u(y1, a)], [u(y2, a)], [u(y3, a)], [u(y4, a)]])

                # Producto punto entre las dos matrices
                dst[j, i, c] = np.dot(np.dot(mat_l, mat_m), mat_r)

    sys.stderr.write('\n')

    # Flushing the buffer
    sys.stderr.flush()
    return dst


x=[] #Crear vector de caracteristicas
res=[]
z=[]
#Factor de escala
ratio = 2
# Coeficiente
a = -1/2

imagesMoto = glob.glob('Moto*.JPG')

for fname in imagesMoto:

  # Leer la imagen
  img = cv2.imread(fname)

  # Calcula histograma
  hsv = cv.cvtColor(img,cv.COLOR_BGR2HSV)
  hist = cv.calcHist( [hsv], [0, 1], None, [7, 8], [0, 180, 0, 256] )
  plt.imshow(hist,interpolation = 'nearest')
  plt.show()

  for i in range(len(hist[0])): #Se recorren las columnas
    for j in range(len(hist)): ##Se recorren las filas
      #print(hist[j][i])
      x.append(hist[j][i]) #Se llena el vector con las características
  x.append('2')

imagesPart = glob.glob('Particular*.JPG')

for fname in imagesPart:

  # Leer la imagen
  img = cv2.imread(fname)

  # Calcula histograma
  hsv = cv.cvtColor(img,cv.COLOR_BGR2HSV)
  hist = cv.calcHist( [hsv], [0, 1], None, [7, 8], [0, 180, 0, 256] )
  plt.imshow(hist,interpolation = 'nearest')
  plt.show()

  for i in range(len(hist[0])): #Se recorren las columnas
```

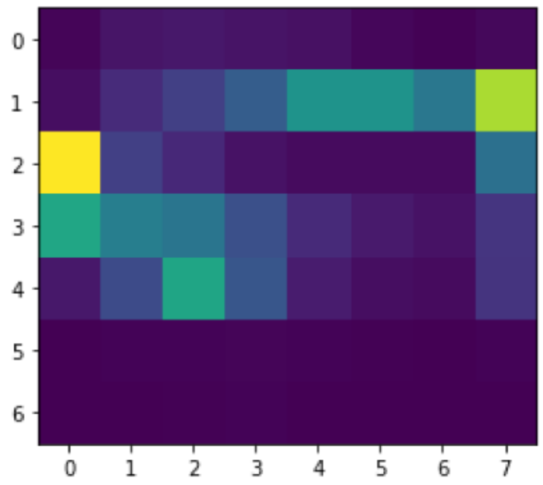
```
      for j in range(len(hist)): ##Se recorren las filas
        #print(hist[j][i])
        x.append(hist[j][i]) #Se llena el vector con las características
    x.append('1')

  imagesPub = glob.glob('Publico*.JPG')

  for fname in imagesPub:

    # Leer la imagen
    img = cv2.imread(fname)

    # Calcula histograma
    hsv = cv.cvtColor(img,cv.COLOR_BGR2HSV)
    hist = cv.calcHist( [hsv], [0, 1], None, [7, 8], [0, 180, 0, 256] )
    plt.imshow(hist,interpolation = 'nearest')
    plt.show()

    for i in range(len(hist[0])): #Se recorren las columnas
      for j in range(len(hist)): ##Se recorren las filas
        #print(hist[j][i])
        x.append(hist[j][i]) #Se llena el vector con las características
    x.append('0')

  res= np.array(x).reshape(len(imagesPart)+len(imagesPub)+len(imagesMoto),(len(hist[0])*len(hist))+1)
  carros=pd.DataFrame(res)
  carros.to_csv('caracteristicas.csv')
```

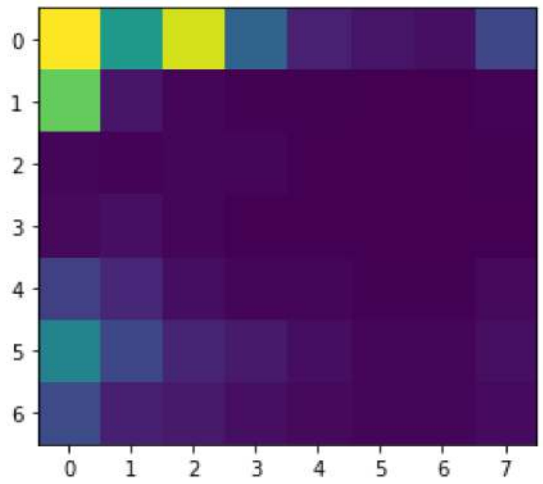

0
1
2
3
4
5
6
0 1 2 3 4 5 6 7

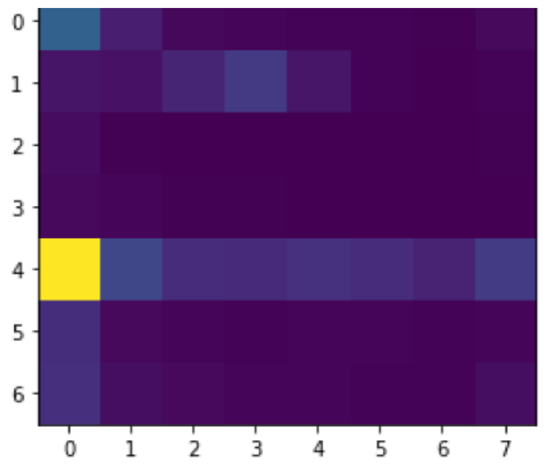

0
1
2
3
4
5
6
0 1 2 3 4 5 6 7

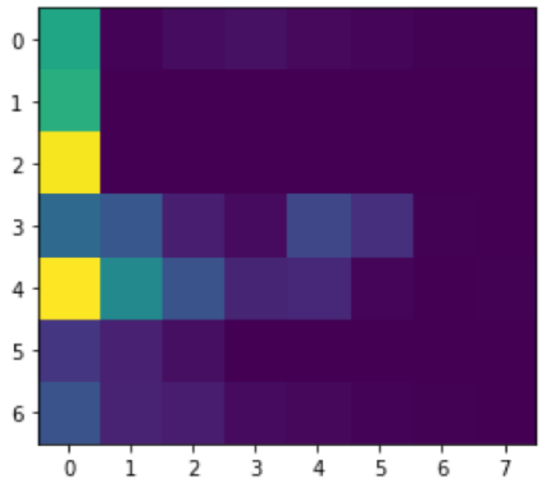

0
1
2
3
4
5
6
0 1 2 3 4 5 6 7

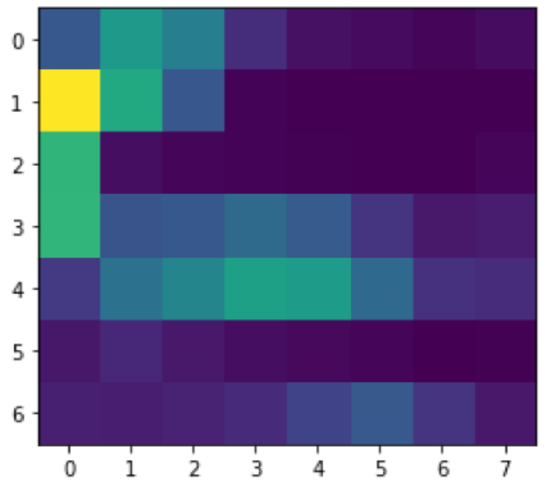

0
1
2
3
4
5
6
0 1 2 3 4 5 6 7

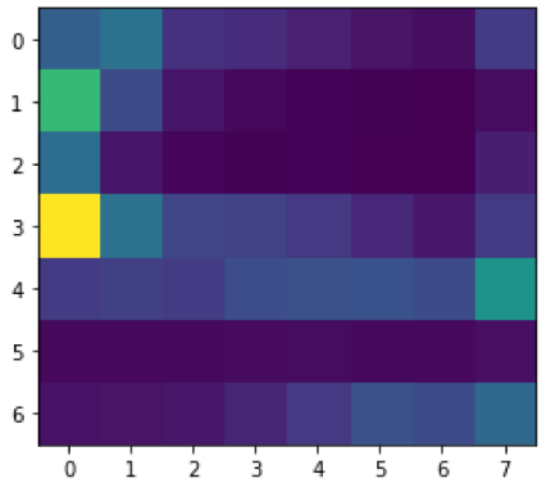

0
1
2
3
4
5
6
0 1 2 3 4 5 6 7

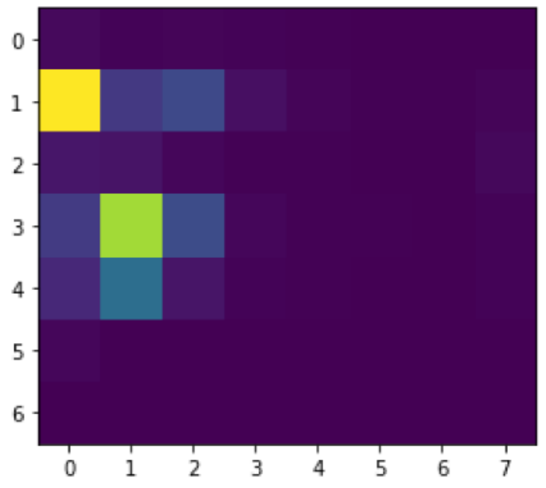

0
1
2
3
4
5
6
0 1 2 3 4 5 6 7

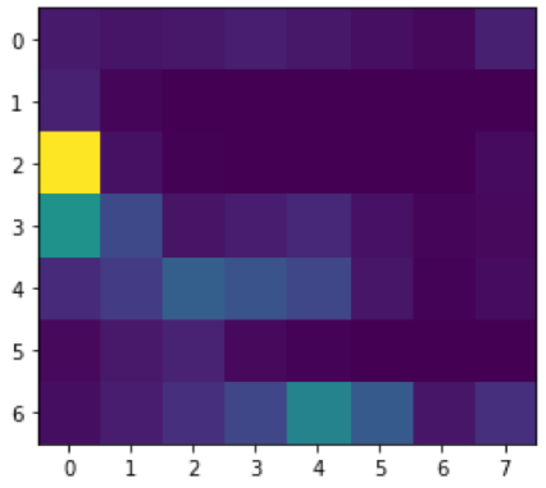

0
1
2
3
4
5
6
0
1
2
3
4
5
6
7


0
1
2
3
4
5
6
0
1
2
3
4
5
6
7

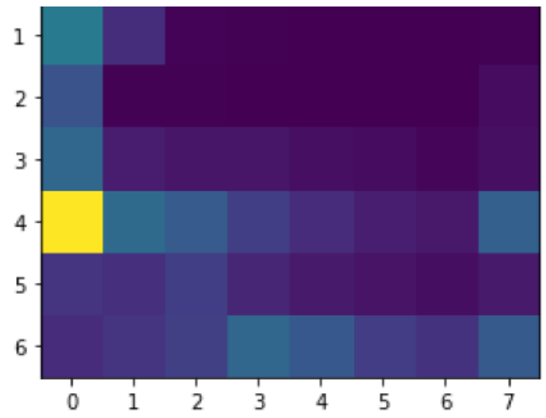

0
1
2
3
4
5
6
0
1
2
3
4
5
6
7

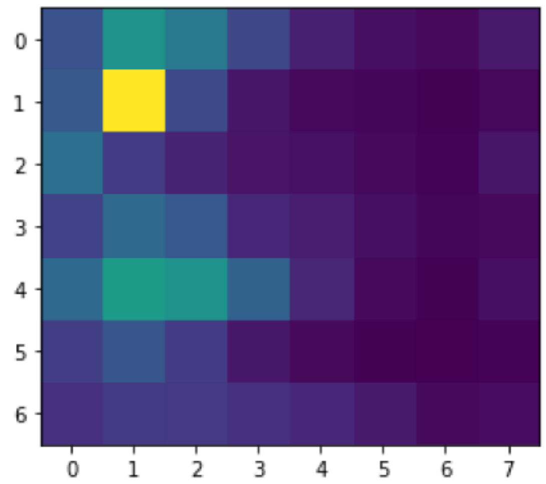

0
1
2
3
4
5
6
0
1
2
3
4
5
6
7

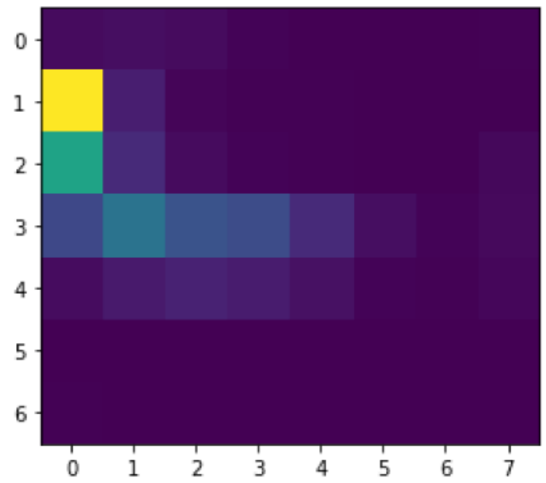

0
1
2
3
4
5
6
0
1
2
3
4
5
6
7

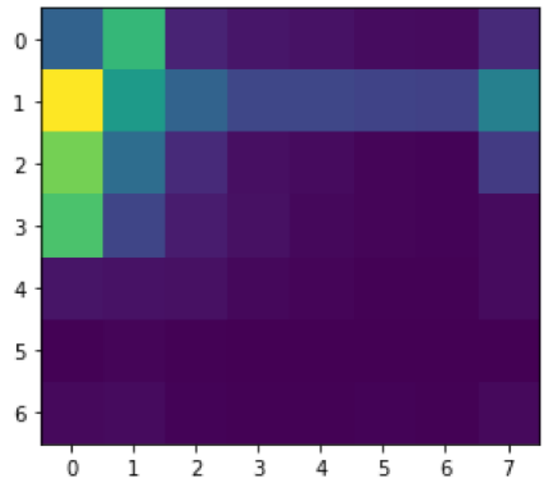

0
1
2
3
4
5
6
0
1
2
3
4
5
6
7

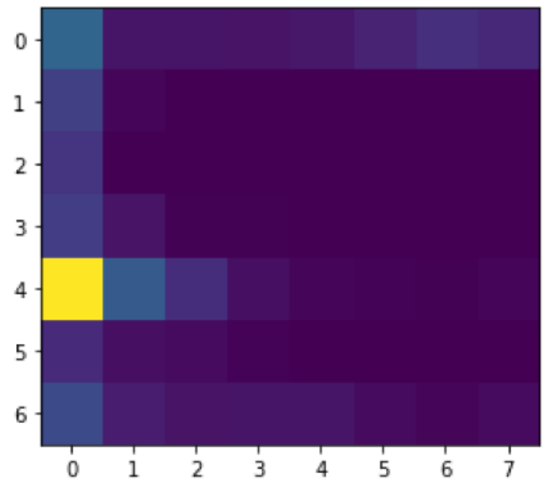

0

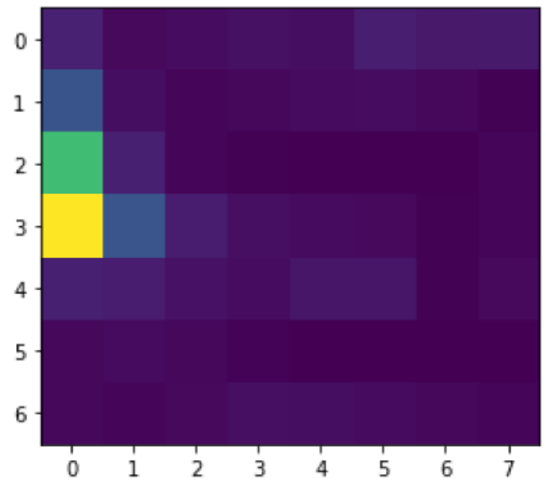

1
2
3
4
5
6
0
1
2
3
4
5
6
7


0
1
2
3
4
5
6
0
1
2
3
4
5
6
7

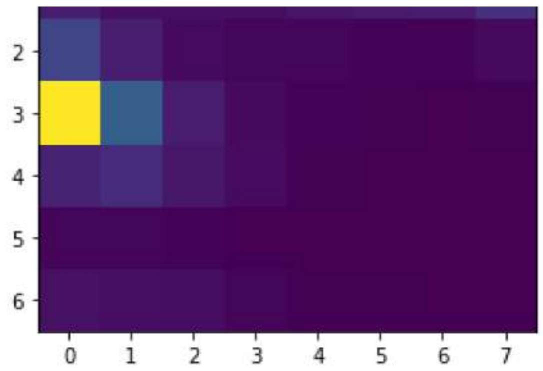

0
1
2
3
4
5
6
0
1
2
3
4
5
6
7

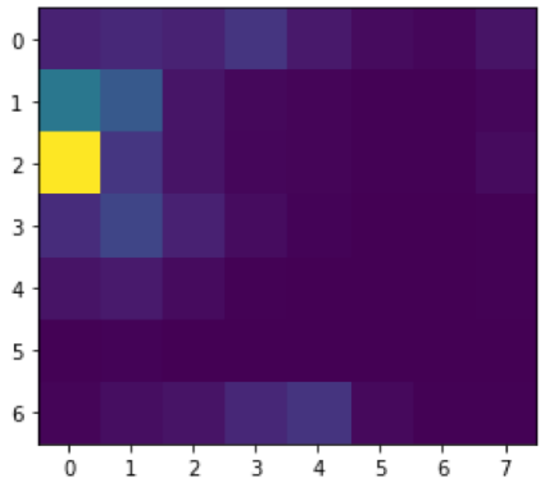

0
1
2
3
4
5
6
0
1
2
3
4
5
6
7

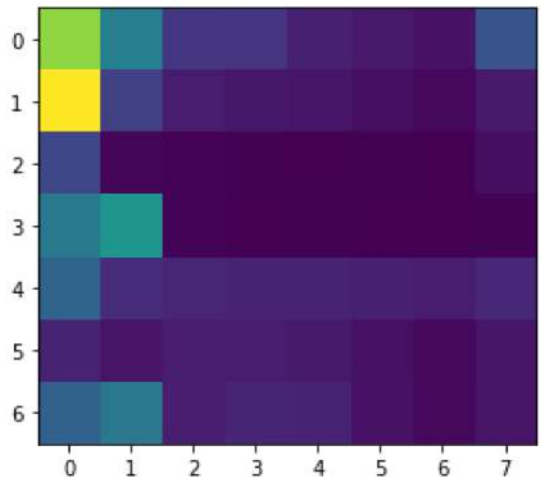

0
1
2
3
4
5
6
0
1
2
3
4
5
6
7

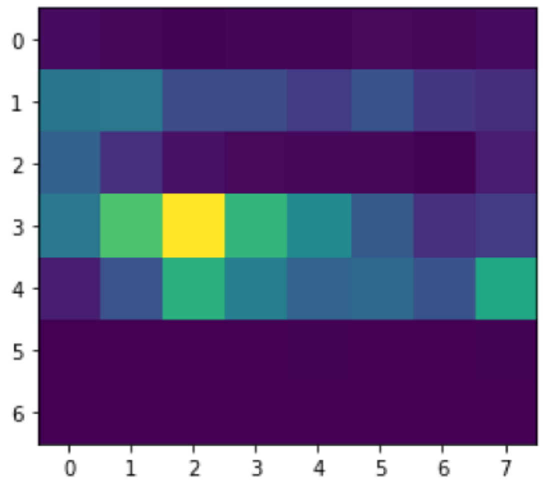

0
1
2
3
4
5
6
0
1
2
3
4
5
6
7

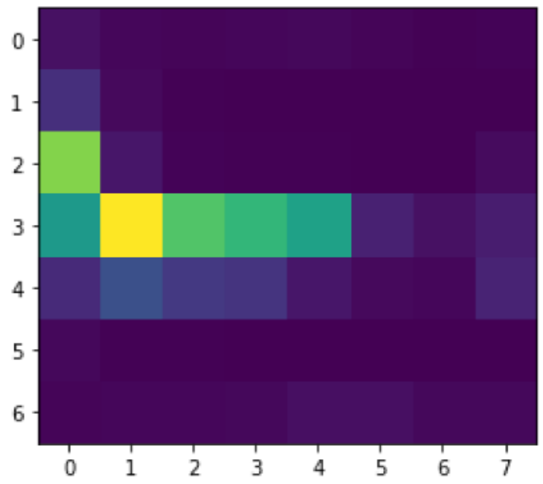

0
1

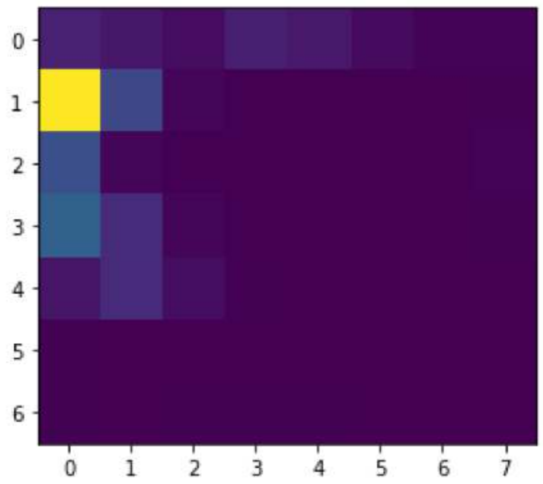

2
3
4
5
6
0 1 2 3 4 5 6 7


0
1
2
3
4
5
6
0 1 2 3 4 5 6 7

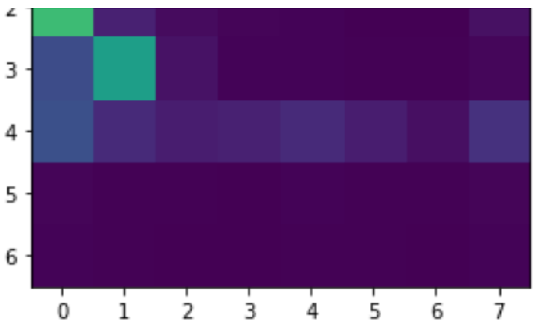

0
1
2
3
4
5
6
0 1 2 3 4 5 6 7

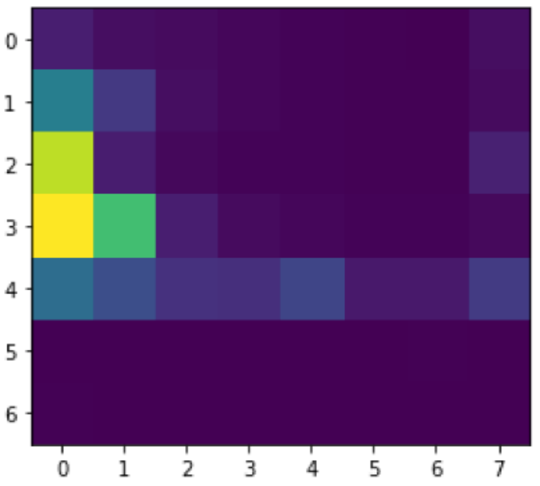

0
1
2
3
4
5
6
0 1 2 3 4 5 6 7

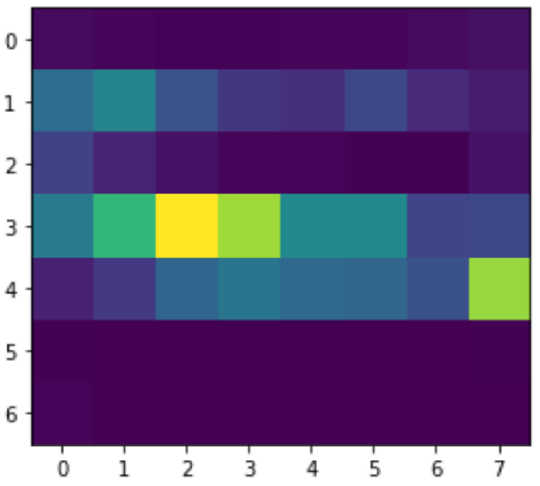

0
1
2
3
4
5
6
0 1 2 3 4 5 6 7

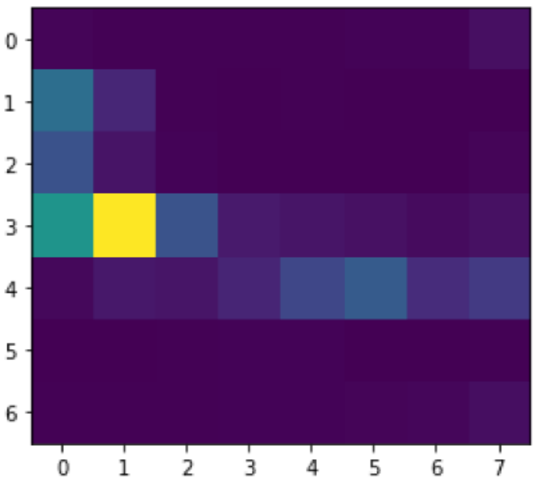

0
1
2
3
4
5
6
0 1 2 3 4 5 6 7

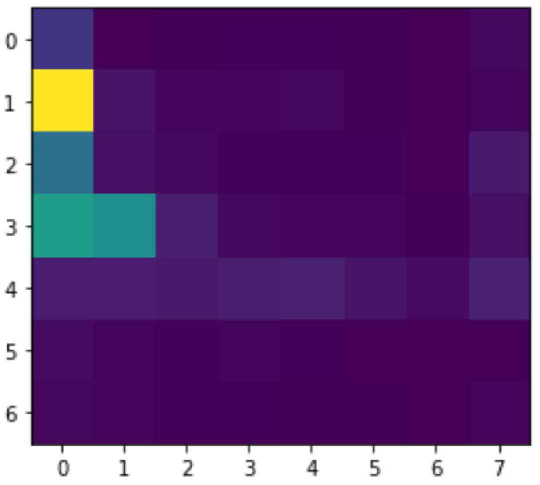

0
1

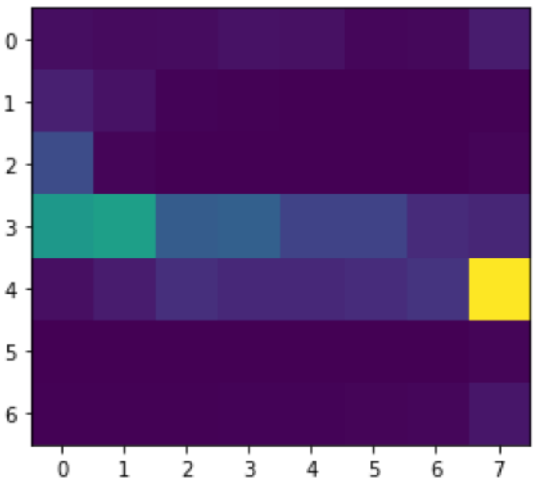

2
3
4
5
6
0 1 2 3 4 5 6 7


0
1
2
3
4
5
6
0 1 2 3 4 5 6 7

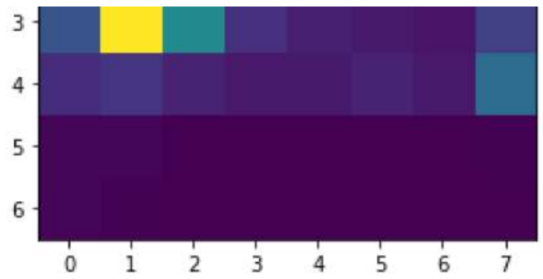

0
1
2
3
4
5
6
0 1 2 3 4 5 6 7

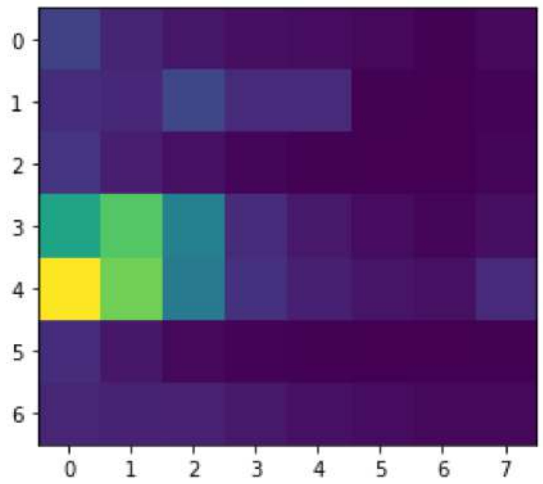

0
1
2
3
4
5
6
0 1 2 3 4 5 6 7

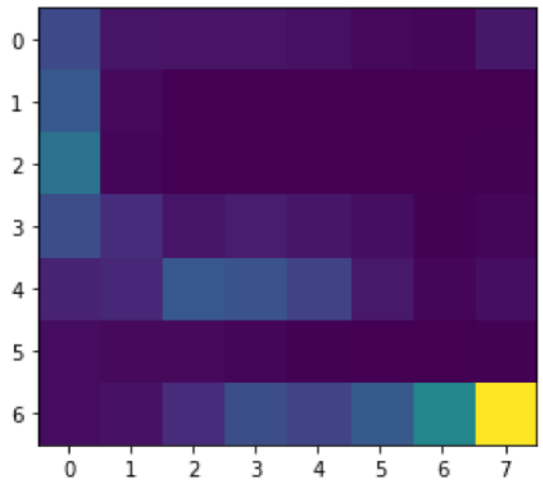

0
1
2
3
4
5
6
0 1 2 3 4 5 6 7

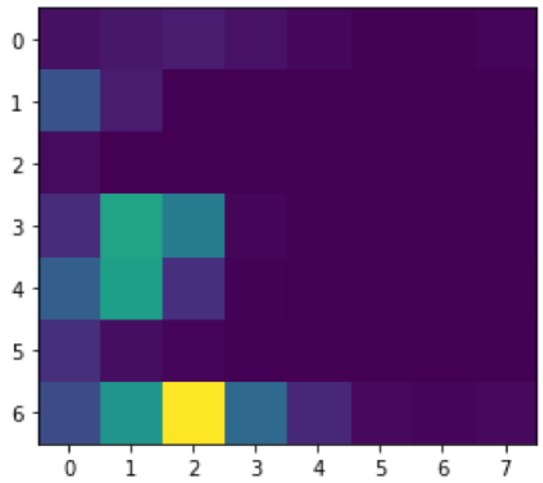

0
1
2
3
4
5
6
0 1 2 3 4 5 6 7

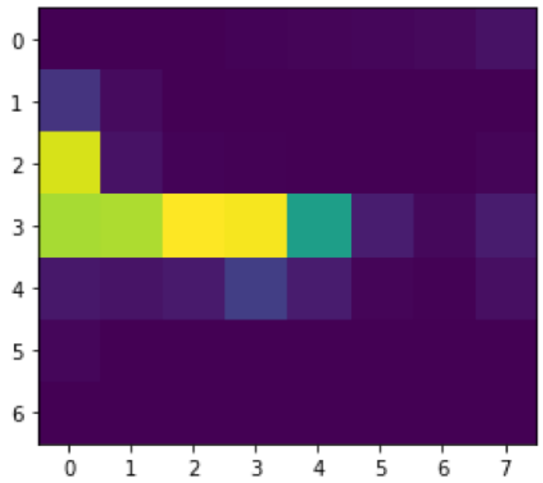

0
1
2

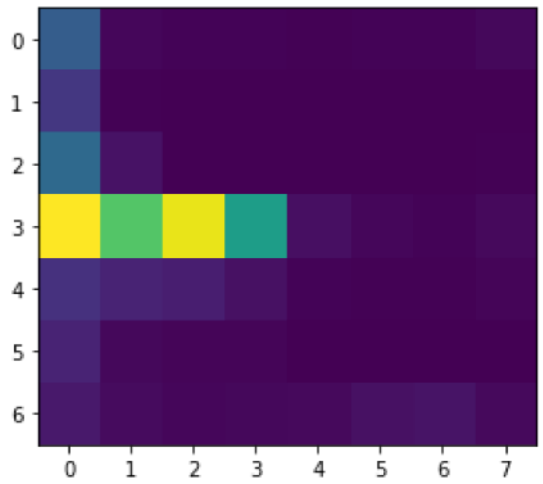

3
4
5
6
0 1 2 3 4 5 6 7


0
1
2
3
4
5
6
0 1 2 3 4 5 6 7

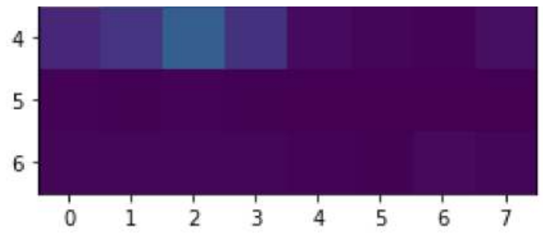

0
1
2
3
4
5
6
0 1 2 3 4 5 6 7

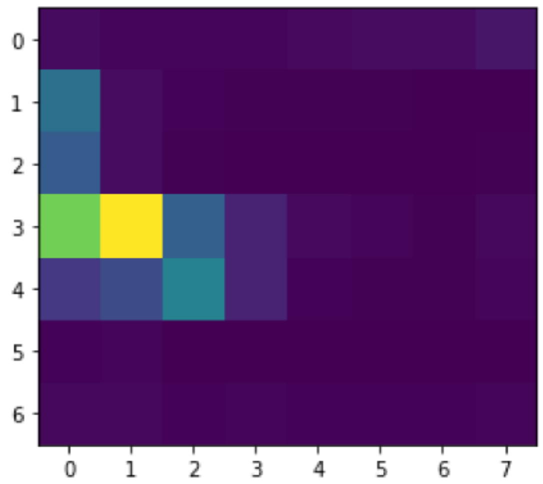

0
1
2
3
4
5
6
0 1 2 3 4 5 6 7

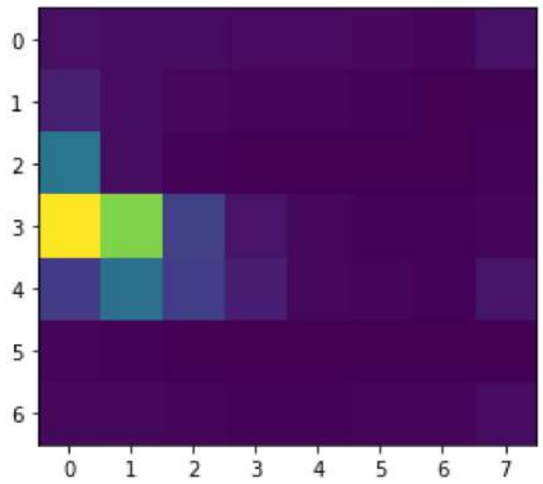

0
1
2
3
4
5
6
0 1 2 3 4 5 6 7

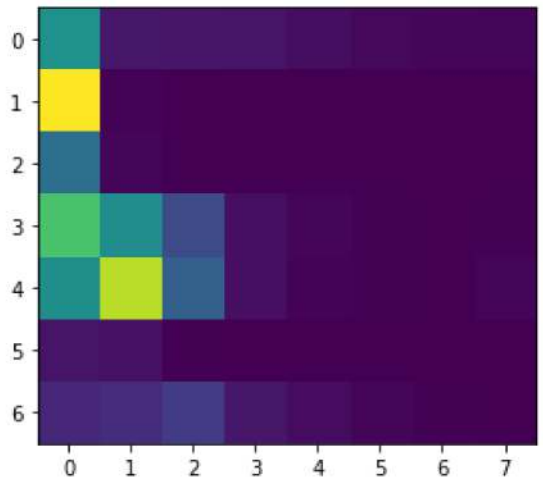

0
1
2
3
4
5
6
0 1 2 3 4 5 6 7

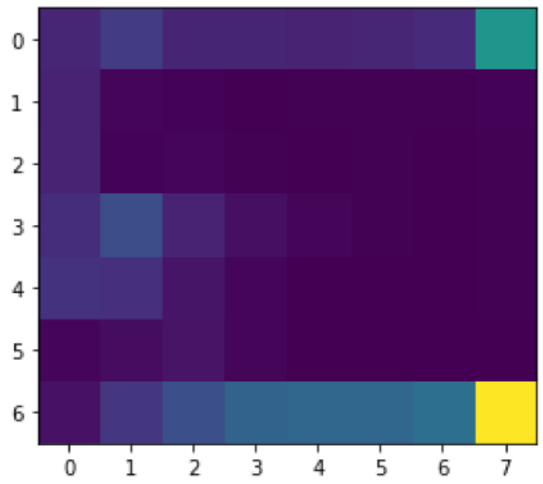

0
1
2
3

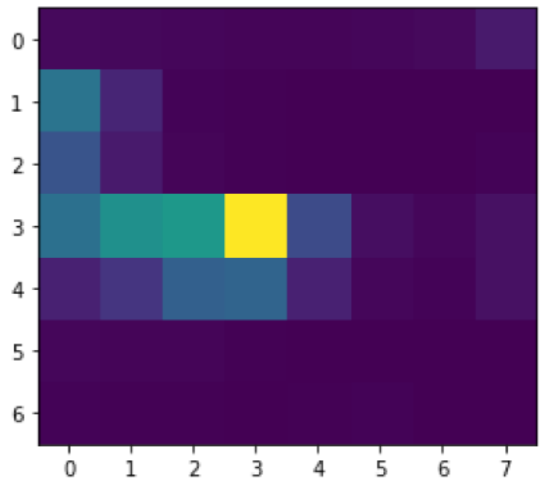

4
5
6
0
1
2
3
4
5
6
7


0
1
2
3
4
5
6
0
1
2
3
4
5
6
7

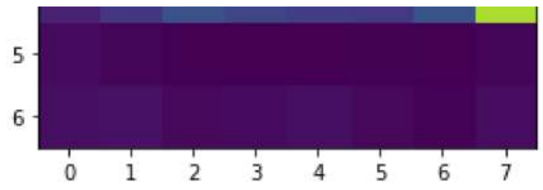

0
1
2
3
4
5
6
0
1
2
3
4
5
6
7

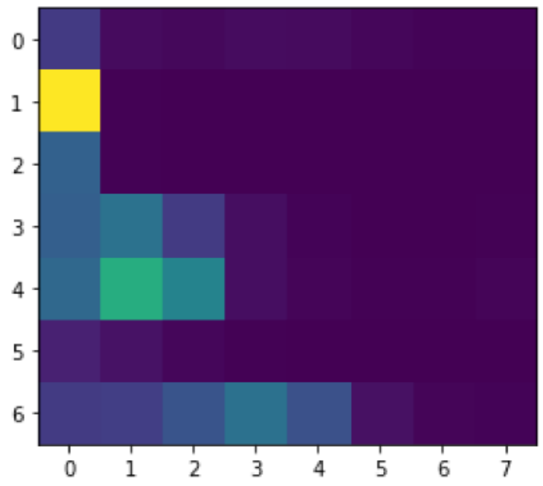

0
1
2
3
4
5
6
0
1
2
3
4
5
6
7

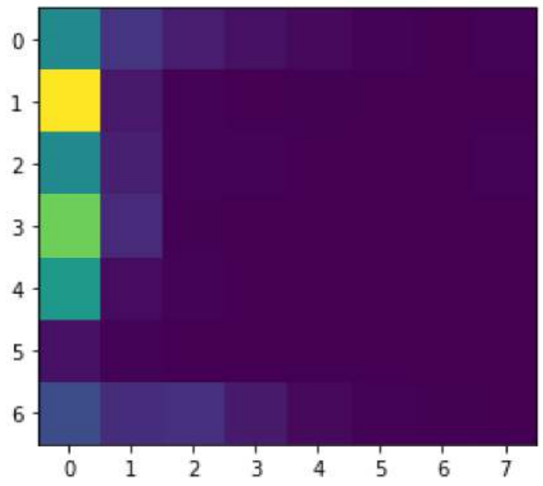

0
1
2
3
4
5
6
0
1
2
3
4
5
6
7

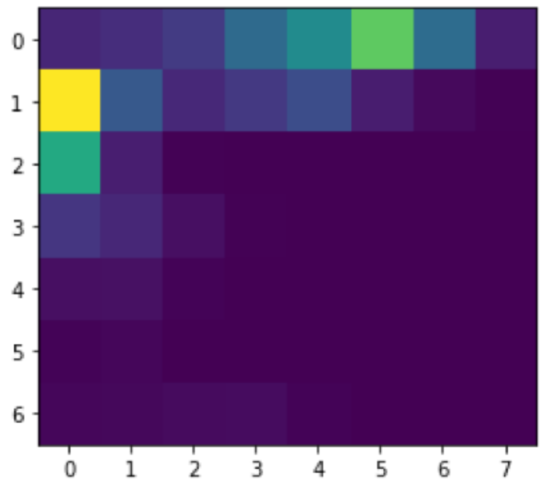

0
1
2
3
4
5
6
0
1
2
3
4
5
6
7

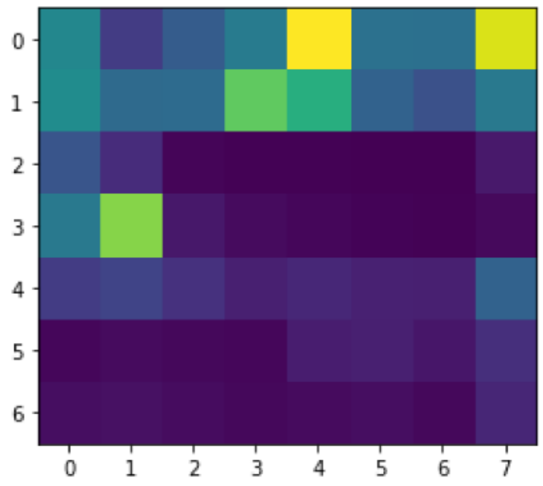

0
1
2
3
4

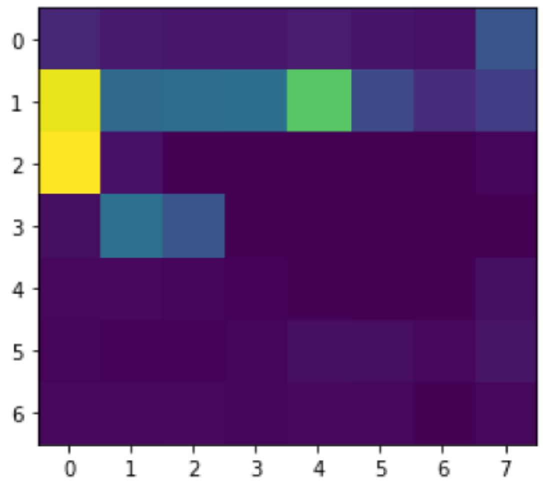

5
6
0
1
2
3
4
5
6
7


0
1
2
3
4
5
6
0
1
2
3
4
5
6
7

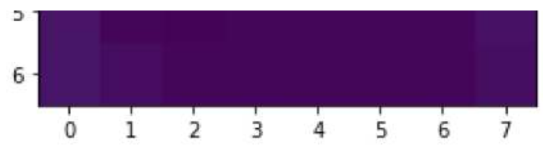

0
1
2
3
4
5
6
0
1
2
3
4
5
6
7

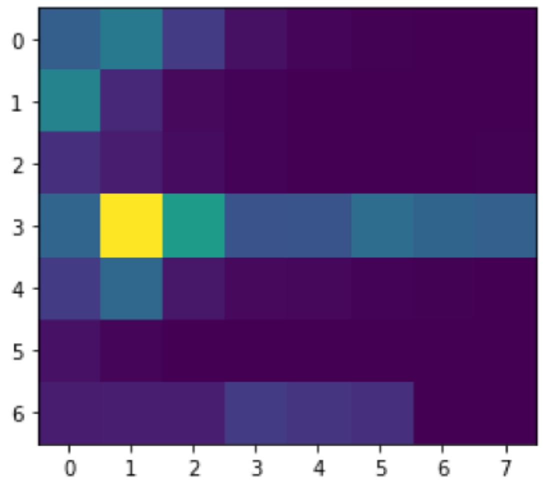

0
1
2
3
4
5
6
0
1
2
3
4
5
6
7

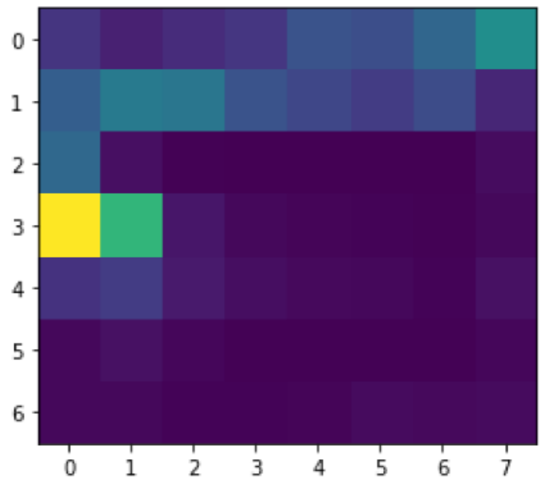

0
1
2
3
4
5
6
0
1
2
3
4
5
6
7

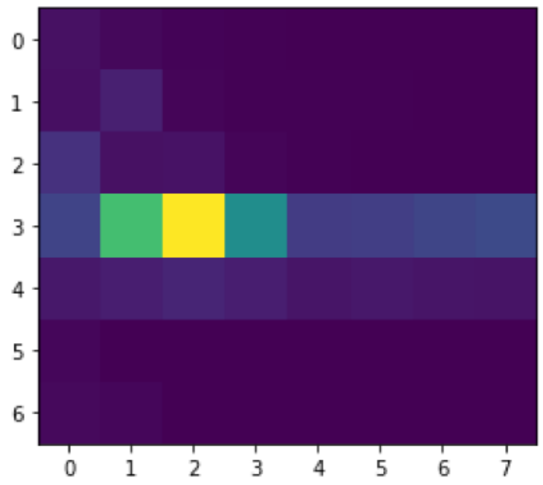

0
1
2
3
4
5
6
0
1
2
3
4
5
6
7

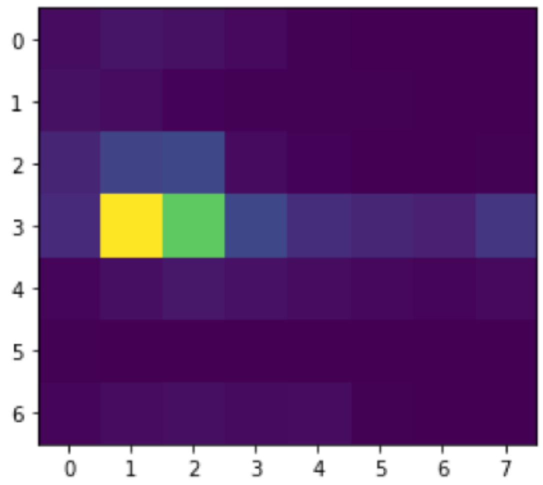

0
1
2
3
4

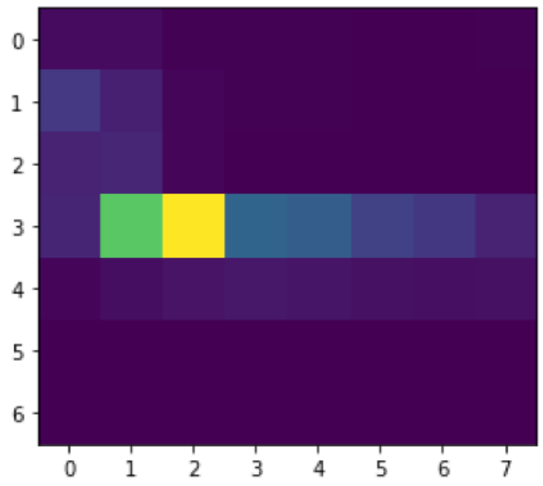

5
6
0 1 2 3 4 5 6 7


0 1 2 3 4 5 6
0 1 2 3 4 5 6 7

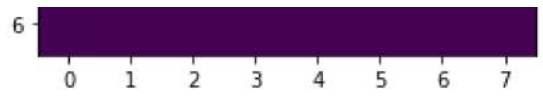

0 1 2 3 4 5 6
0 1 2 3 4 5 6 7

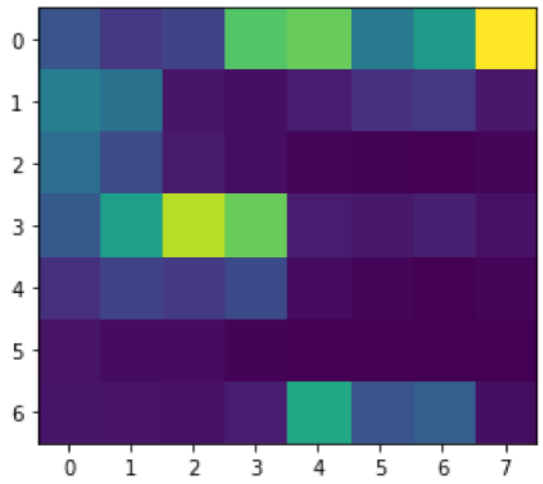

0 1 2 3 4 5 6
0 1 2 3 4 5 6 7

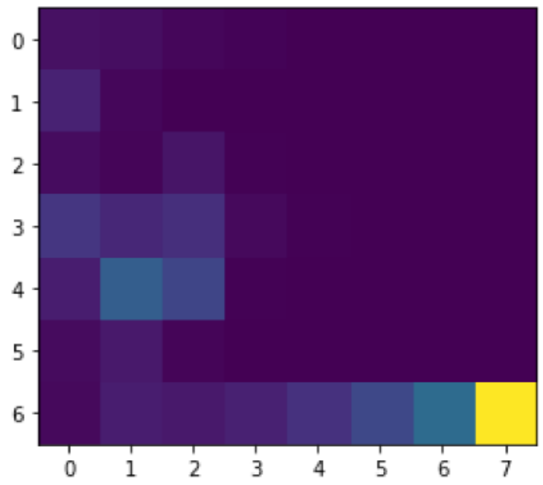

0 1 2 3 4 5 6
0 1 2 3 4 5 6 7

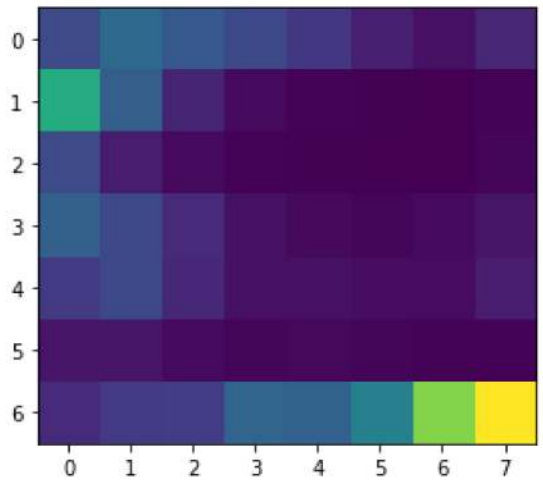

0 1 2 3 4 5 6
0 1 2 3 4 5 6 7

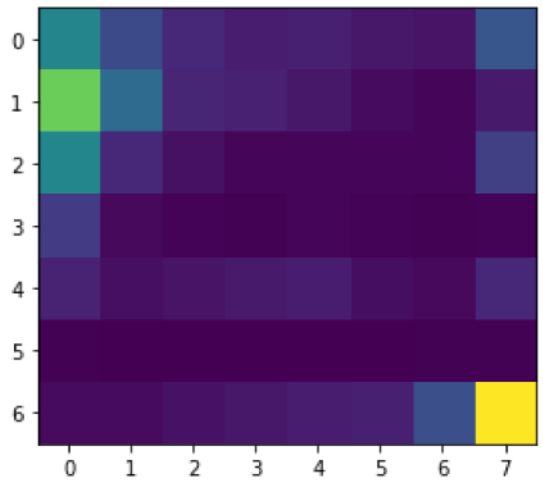

0 1 2 3 4 5

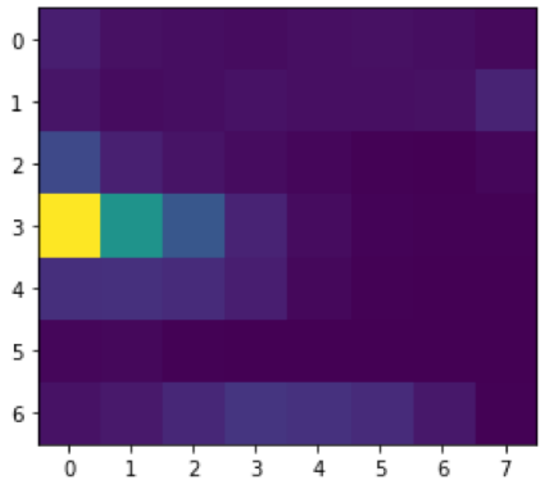

6
0 1 2 3 4 5 6 7


0 1 2 3 4 5 6
0 1 2 3 4 5 6 7

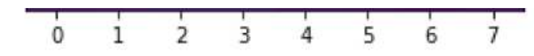

0 1 2 3 4 5 6
0 1 2 3 4 5 6 7

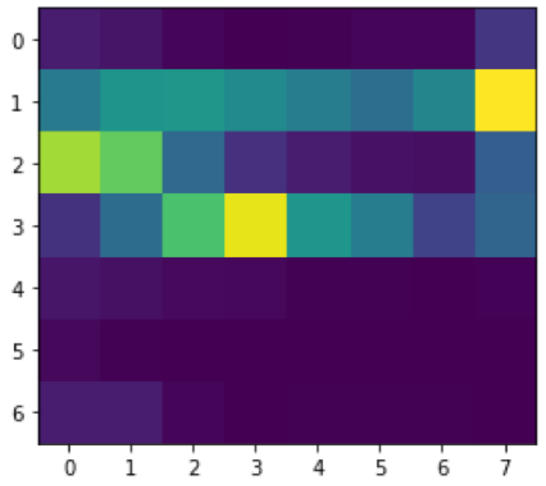

0 1 2 3 4 5 6
0 1 2 3 4 5 6 7


0 1 2 3 4 5 6
0 1 2 3 4 5 6 7


0 1 2 3 4 5 6
0 1 2 3 4 5 6 7


0 1 2 3 4 5 6





```
clases_carros = pd.read_csv('caracteristicas.csv')
x = clases_carros.iloc[:,:-1].values
y = clases_carros.iloc[:,-1].values
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.3, random_state = 21)
################################Decision tree##################################
tree = DecisionTreeClassifier(criterion = "entropy", random_state = 21)
#Train
tree.fit(x_train,y_train)

y_pred = tree.predict(x_test)
print(y_pred)
treeConfMat = confusion_matrix(y_test,y_pred)
sns.heatmap(treeConfMat,annot=True)
plt.xlabel("Predicted label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show
print("accuracy")
print(accuracy_score(y_test,y_pred))
```

```
[2 1 2 1 0 2 2 1 1 1 2 2 1 0 1 0 1 1 1 0]
accuracy
1.0
```