Instituto Tecnológico de Aeronáutica - ITA
Inteligência Artificial para Robótica Móvel - CT-213
Aluno: Vinícius José de Menezes Pereira

Relatório do Laboratório 9 - Detecção de Objetos

1. Breve Explicação em Alto Nível da Implementação

Primeira parte:

Na primeira parte, foi construída uma rede neural de 9 camadas conforme foi pedido pelo roteiro. Nas camadas, foram utilizadas convoluções 2D, Batch Normalization, MaxPooling 2D e LeakyReLu como função de ativação. De modo particular, a camada 7 foi dividida em 2 partes, 7A e 7B, sendo 7B uma skip conection. Isso é uma característica das residual networks, onde a informação da camada 6 é de certa forma privilegiada, pois ao concatenarmos a camada 8 com a 7B, privilegiamos a informação anterior. O esquema das camadas é mostrado na figura a seguir

Segunda parte:

Na segunda parte, seguiu-se o modelo proposto da YoloDetector.

Descrição das funções implementadas:

```
def detect(self, image):
```

Função de detecção principal, que será chamada na parte de teste. Primeiramente processa a imagem com `preprocess_image()`, depois passa a imagem pela rede neural e faz a detecção com `process_yolo_output()`,selecionando os postes e a bola.

```
def preprocess_image(self, image):
```

Função de processamento da imagem. Primeiramente diminui a resolução da imagem em 4 vezes, indo de 640x480 para 160x120. Depois transforma a imagem num vetor do numpy para normalizar os valores encontrados em cada pixel. Após isso, redimensiona a imagem para o formato adequado de `(1, 120, 160, 3)`, que será utilizado.

```
def process_yolo_output(self, output):
```

Dado o output da rede neural que fez sua predição na imagem, seleciona local e as bordas da bola e das traves. Para a bola, faz isso selecionando o local de maior probabilidade de achá-la. Para as traves, seleciona os primeiros e segundos melhores locais onde há probabilidade de haver trave para serem as traves.

1.1. Sumário do Modelo

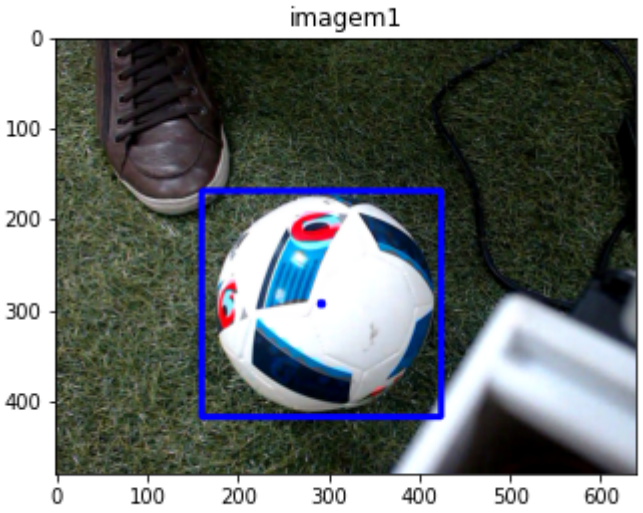
```
Model: "ITA_YOLO"
_________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #     Connected to
=================================================================================================
 input_2 (InputLayer)        [(None, 120, 160, 3       0           []
                             )]

 conv_1 (Conv2D)             (None, 120, 160, 8)       216         ['input_2[0][0]']
```

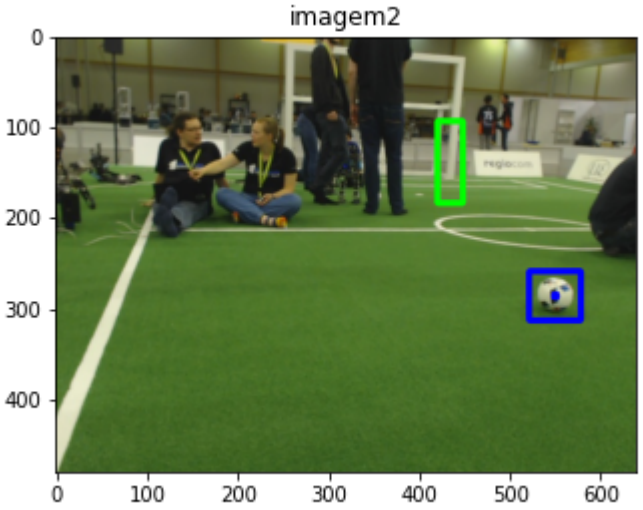
```
norm_1 (BatchNormalization)    (None, 120, 160, 8)  32          ['conv_1[0][0]']

leaky_relu_1 (LeakyReLU)       (None, 120, 160, 8)  0           ['norm_1[0][0]']

conv_2 (Conv2D)                (None, 120, 160, 8)  576         ['leaky_relu_1[0][0]']

norm_2 (BatchNormalization)    (None, 120, 160, 8)  32          ['conv_2[0][0]']

leaky_relu_2 (LeakyReLU)       (None, 120, 160, 8)  0           ['norm_2[0][0]']

conv_3 (Conv2D)                (None, 120, 160, 16  1152        ['leaky_relu_2[0][0]']
                               )

norm_3 (BatchNormalization)    (None, 120, 160, 16  64          ['conv_3[0][0]']
                               )

leaky_relu_3 (LeakyReLU)       (None, 120, 160, 16  0           ['norm_3[0][0]']
                               )

max_pool_3 (MaxPooling2D)      (None, 120, 160, 16  0           ['leaky_relu_3[0][0]']
                               )

conv_4 (Conv2D)                (None, 120, 160, 32  4608        ['max_pool_3[0][0]']
                               )

norm_4 (BatchNormalization)    (None, 120, 160, 32  128         ['conv_4[0][0]']
                               )

leaky_relu_4 (LeakyReLU)       (None, 120, 160, 32  0           ['norm_4[0][0]']
                               )

max_pool_4 (MaxPooling2D)      (None, 120, 160, 32  0           ['leaky_relu_4[0][0]']
                               )

conv_5 (Conv2D)                (None, 120, 160, 64  18432       ['max_pool_4[0][0]']
                               )

norm_5 (BatchNormalization)    (None, 120, 160, 64  256         ['conv_5[0][0]']
                               )

leaky_relu_5 (LeakyReLU)       (None, 120, 160, 64  0           ['norm_5[0][0]']
                               )

max_pool_5 (MaxPooling2D)      (None, 120, 160, 64  0           ['leaky_relu_5[0][0]']
                               )

conv_6 (Conv2D)                (None, 120, 160, 64  36864       ['max_pool_5[0][0]']
                               )

norm_6 (BatchNormalization)    (None, 120, 160, 64  256         ['conv_6[0][0]']
                               )

leaky_relu_6 (LeakyReLU)       (None, 120, 160, 64  0           ['norm_6[0][0]']
                               )

max_pool_6 (MaxPooling2D)      (None, 120, 160, 64  0           ['leaky_relu_6[0][0]']
                               )

conv_7 (Conv2D)                (None, 120, 160, 12  73728       ['max_pool_6[0][0]']
                               8)

norm_7 (BatchNormalization)    (None, 120, 160, 12  512         ['conv_7[0][0]']
                               8)

leaky_relu_7 (LeakyReLU)       (None, 120, 160, 12  0           ['norm_7[0][0]']
                               8)

conv_skip (Conv2D)             (None, 120, 160, 12  8192        ['max_pool_6[0][0]']
                               8)

conv_8 (Conv2D)                (None, 120, 160, 25  294912      ['leaky_relu_7[0][0]']
                               6)

norm_skip (BatchNormalization)  (None, 120, 160, 12  512        ['conv_skip[0][0]']
                               8)

norm_8 (BatchNormalization)    (None, 120, 160, 25  1024        ['conv_8[0][0]']
                               6)

leaky_relu_skip (LeakyReLU)    (None, 120, 160, 12  0           ['norm_skip[0][0]']
                               8)
```

```
leaky_relu_8 (LeakyReLU)        (None, 120, 160, 25  0           ['norm_8[0][0]']
                                6)

concat (Concatenate)            (None, 120, 160, 38  0           ['leaky_relu_skip[0][0]',
                                4)                                'leaky_relu_8[0][0]']

conv_9 (Conv2D)                 (None, 120, 160, 10  3850        ['concat[0][0]']
                                )

==================================================================================================
Total params: 445,346
Trainable params: 443,938
Non-trainable params: 1,408
__________________________________________________________________________________________________
```

2. Figuras Comprovando Funcionamento do Código

2.1. Detecção de Objetos com YOLO





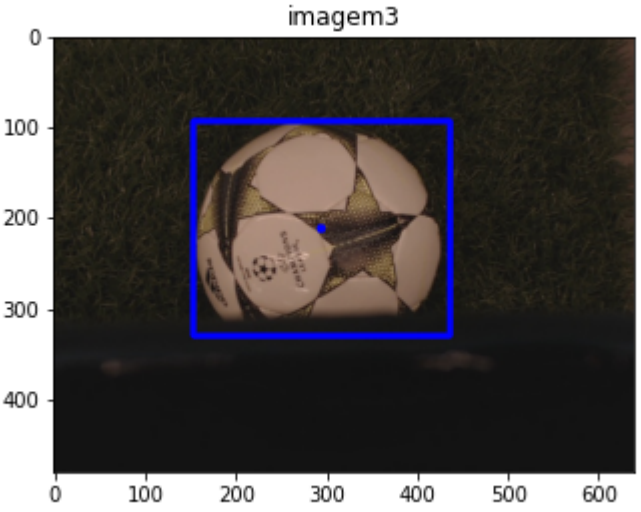


imagem3
0
100
200
300
400
0
100
200
300
400
500
600

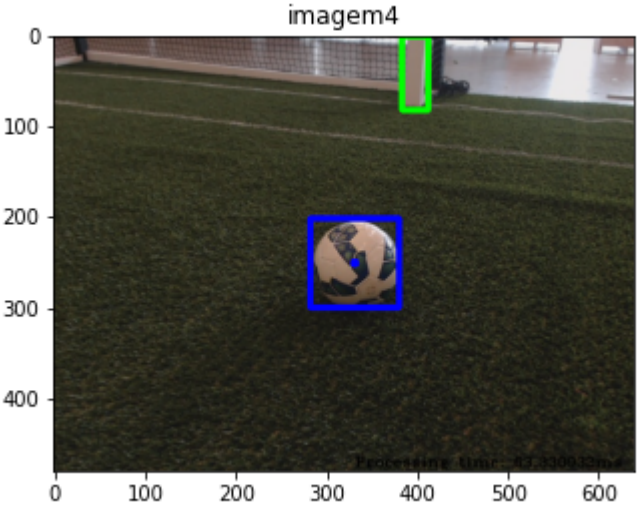


imagem4
0
100
200
300
400
0
100
200
300
400
500
600

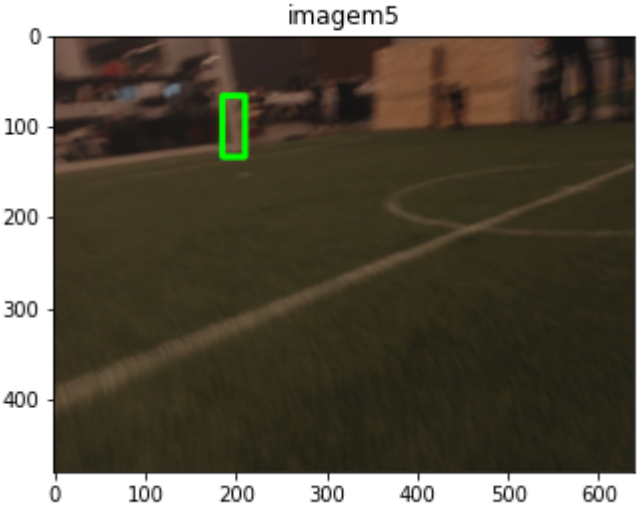


imagem5
0
100
200
300
400
0
100
200
300
400
500
600

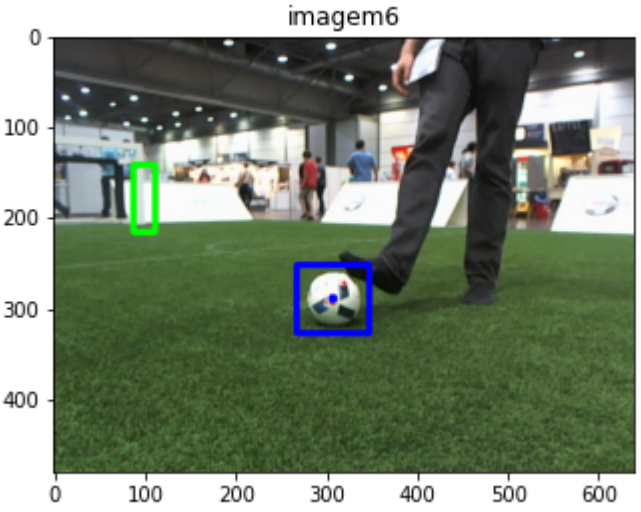


imagem6
0
100
200
300
400
0
100
200
300
400
500
600

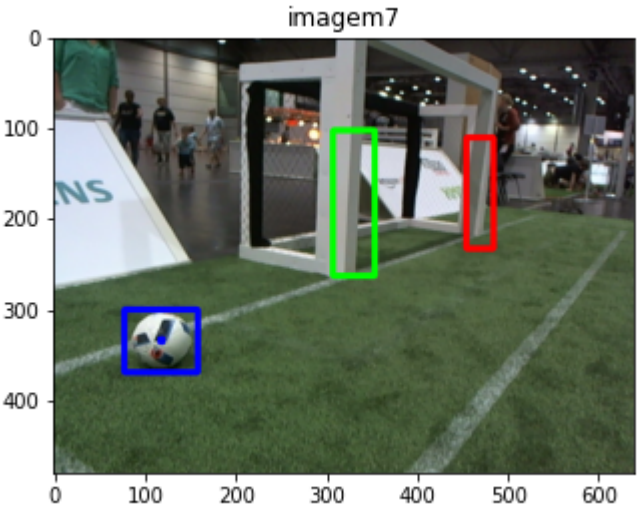


imagem7
0
100
200
300
400
0
100
200
300
400
500
600

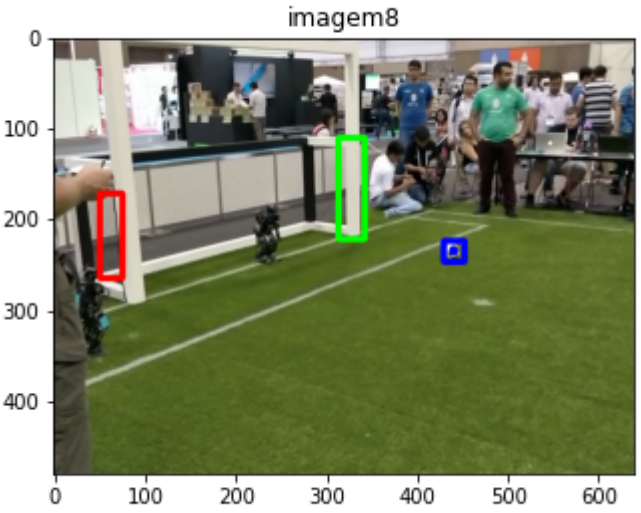
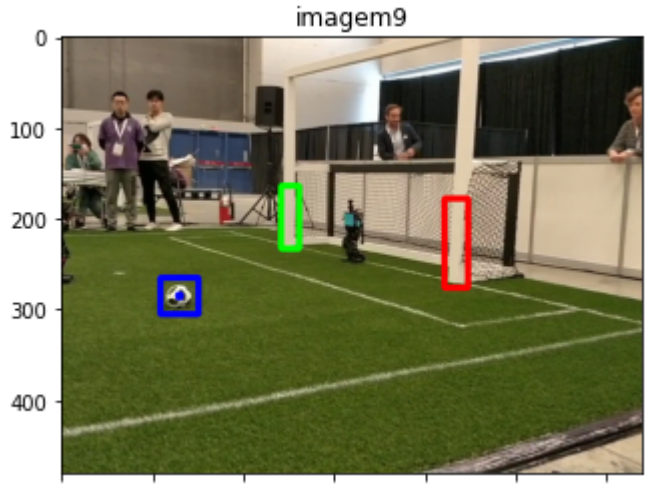
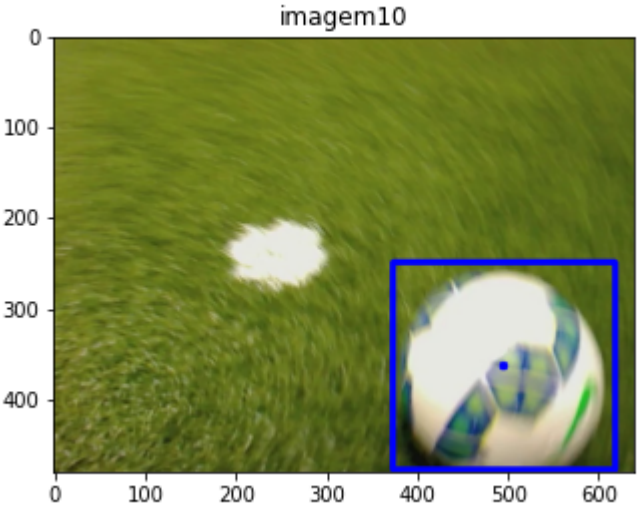


imagem8
0
100
200
300
400
0
100
200
300
400
500
600





3. Discussão

Vemos que o Yolo conseguiu um ótimo desempenho nas imagens propostas. Mesmo em casos em que haviam vários objetos que poderiam causar confusão, como é o caso da marcação do pênalti na imagem 10 ou humanos no meio da foto como em 9, podemos ver um ótimo desempenho. Conseguiu também bom desempenho em imagens borradas como a 9 e a 5, o que é bom, pois a maioria das fotos que o robô irá lidar num jogo real serão assim. Além disso, percebeu bem os locais das traves em situações difíceis como a imagem 6, na imagem 5, que está borrada, e na imagem 4, em que a trave está cortada.