Laboratório 9 - Detecção de Objetos

Marcelo Buga Martins da Silva

CT-213 - Professor Marcos Ricardo Omena de Albuquerque Máximo

02/06/2021

1 Introdução

O propósito desse laboratório foi a implementação do algoritmo YOLO, baseado em Rede Neural Convolucional para a Detecção de Objetos. Esse algoritmo foi utilizado no contexto de futebol de robôs para localizar as traves e a bola em imagens.

Esse relatório apresentará uma breve descrição da implementação da rede neural via Keras, bem como a implementação da detecção dos objetos a partir da saída da rede neural (já treinada).

2 Implementação da rede neural

A implementação dessa rede neural foi bem simples, visto que o Keras a torna muito intuitiva. Basicamente, bastou-se adicionar as camadas de acordo com a arquitetura da rede com as funções Conv2D, BatchNormalization, LeakyReLu e MaxPooling2D, tendo sido usada a função *concatenate* para a *skip connection* realizada. As Figuras 1 representa a saída do *model summary* da rede criada:

```
Model: "ITA_YOLO"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #   Connected to
=================================================================================================
input_1 (InputLayer)         [(None, 120, 160, 3)      0
_________________________________________________________________________________________________
conv_1 (Conv2D)              (None, 120, 160, 8)       216       input_1[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, 60, 80, 16)        0         leaky_relu_3[0][0]
_________________________________________________________________________________________________
conv_4 (Conv2D)              (None, 60, 80, 32)        4608      max_pool_3[0][0]
_________________________________________________________________________________________________
norm_4 (BatchNormalization)  (None, 60, 80, 32)        128       conv_4[0][0]
_________________________________________________________________________________________________
leaky_relu_4 (LeakyReLU)     (None, 60, 80, 32)        0         norm_4[0][0]
_________________________________________________________________________________________________
max_pool_4 (MaxPooling2D)    (None, 30, 40, 32)        0         leaky_relu_4[0][0]
_________________________________________________________________________________________________
conv_5 (Conv2D)              (None, 30, 40, 64)        18432     max_pool_4[0][0]
_________________________________________________________________________________________________
norm_5 (BatchNormalization)  (None, 30, 40, 64)        256       conv_5[0][0]
```

```
leaky_relu_5 (LeakyReLU)        (None, 30, 40, 64)   0           norm_5[0][0]
max_pool_5 (MaxPooling2D)       (None, 15, 20, 64)   0           leaky_relu_5[0][0]
conv_6 (Conv2D)                 (None, 15, 20, 64)   36864       max_pool_5[0][0]
norm_6 (BatchNormalization)     (None, 15, 20, 64)   256         conv_6[0][0]
leaky_relu_6 (LeakyReLU)        (None, 15, 20, 64)   0           norm_6[0][0]
max_pool_6 (MaxPooling2D)       (None, 15, 20, 64)   0           leaky_relu_6[0][0]
conv_7 (Conv2D)                 (None, 15, 20, 128)  73728       max_pool_6[0][0]
norm_7 (BatchNormalization)     (None, 15, 20, 128)  512         conv_7[0][0]
leaky_relu_7 (LeakyReLU)        (None, 15, 20, 128)  0           norm_7[0][0]
conv_skip (Conv2D)              (None, 15, 20, 128)  8192        max_pool_6[0][0]
conv_8 (Conv2D)                 (None, 15, 20, 256)  294912      leaky_relu_7[0][0]
norm_skip (BatchNormalization)  (None, 15, 20, 128)  512         conv_skip[0][0]
norm_8 (BatchNormalization)     (None, 15, 20, 256)  1024        conv_8[0][0]
leaky_relu_skip (LeakyReLU)     (None, 15, 20, 128)  0           norm_skip[0][0]
leaky_relu_8 (LeakyReLU)        (None, 15, 20, 256)  0           norm_8[0][0]
concat (Concatenate)            (None, 15, 20, 384)  0           leaky_relu_skip[0][0]
                                                                 leaky_relu_8[0][0]
conv_9 (Conv2D)                 (None, 15, 20, 10)   3850        concat[0][0]
Total params: 445,346
Trainable params: 443,938
Non-trainable params: 1,408
```

Figura 1: Saída do *model summary* para a Rede Neural implementada

3 Implementação da Detecção de Objetos com YOLO

Para a detecção dos objetos, foi utilizada a função *detect* que chama as funções *preprocess_image* para um pré processamento da imagem, a utilização de YOLO para obtenção do vetor de *output* e *process_yolo_output* para o processamento da resposta da rede neural para cada imagem. Em suma, as funções auxiliares foram assim implementadas:

3.1 preprocess_image

Essa função redimensionou a imagem para a resolução de 160x120 pela biblioteca OpenCV. Em seguida os valores de cor foram normalizados para serem *floats* entre 0 e 1. Por fim, os valores foram redimensionados para um *array* da biblioteca NumPy de forma a serem entrada válida para a rede neural.

3.2 process_yolo_output

Para essa função, tomou-se o *array* de saída da rede neural e avaliou-se o vetor de *features* para cada célula. Primeiro, encontrou-se a célula com a maior probabilidade de ter uma bola, calculando-se os parâmetros da altura e largura do bounding box e a posição de seu centro, conforme a YOLO V2. Fez-se a mesma coisa para as duas maiores probabilidades da presença da trave.

Essa implementação foi bem sucedida, tendo sido a detecção dos objetos bem precisa na maioria dos casos. As Figuras 2, 3, 4, 5 e 6 apresentam resultados da implementação:

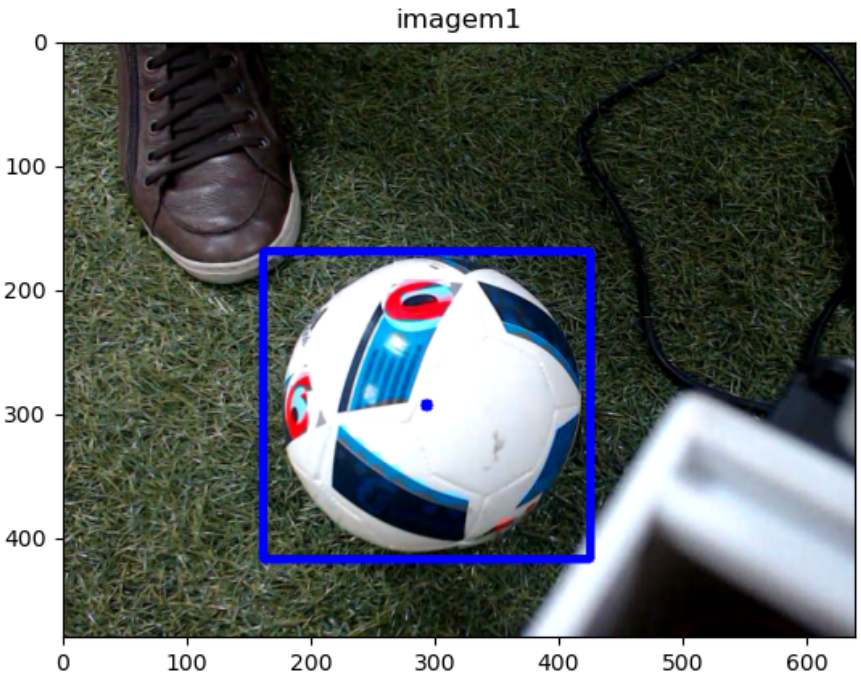

Figura 2: Detecção da Rede Neural para a imagem 1

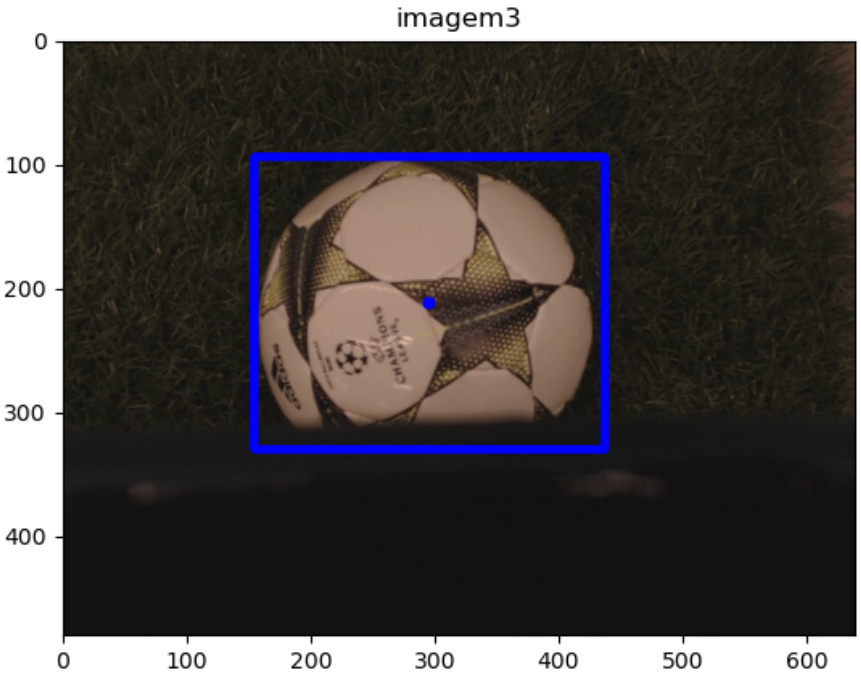

Figura 3: Detecção da Rede Neural para a imagem 3

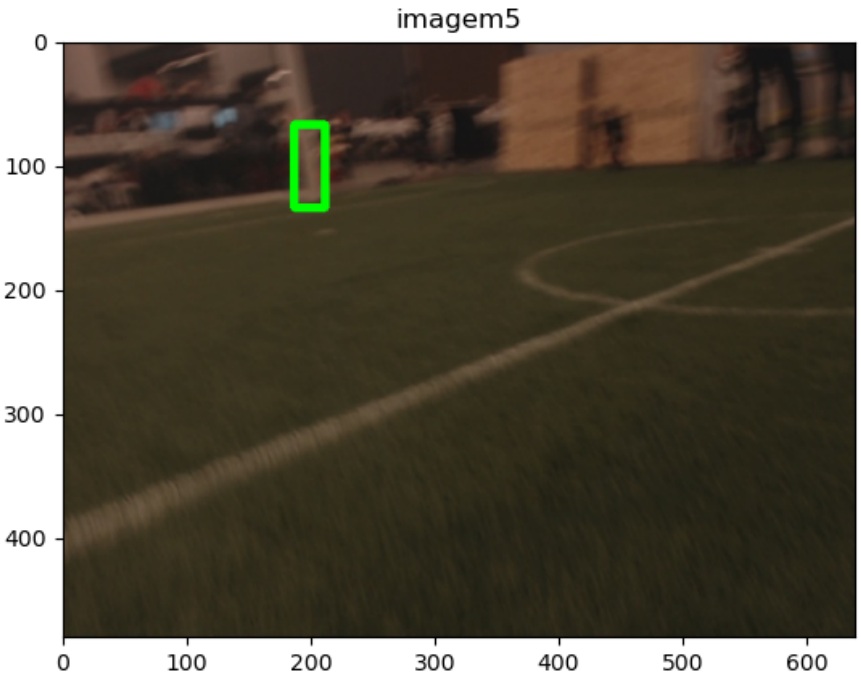

Figura 4: Detecção da Rede Neural para a imagem 5

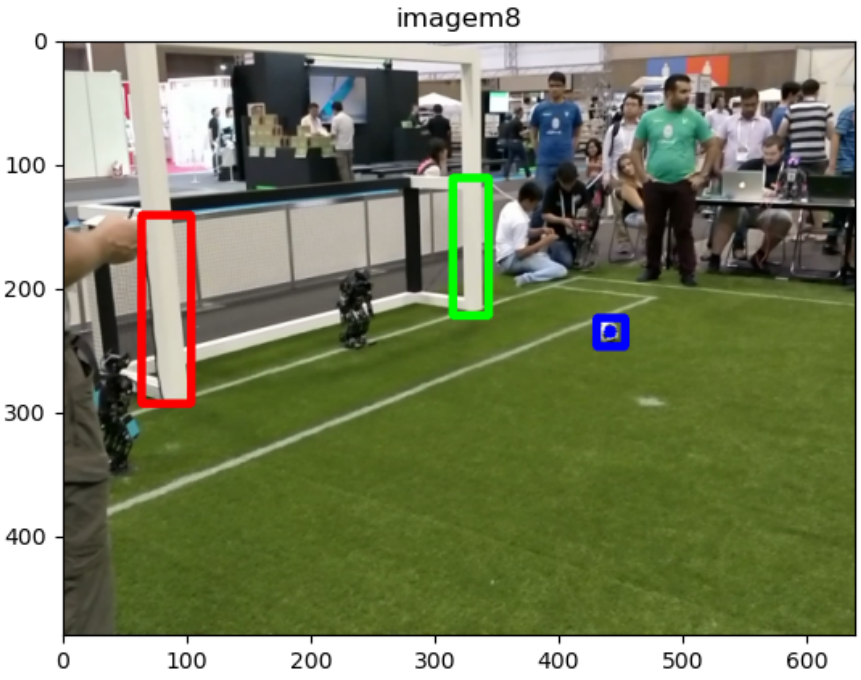

Figura 5: Detecção da Rede Neural para a imagem 8

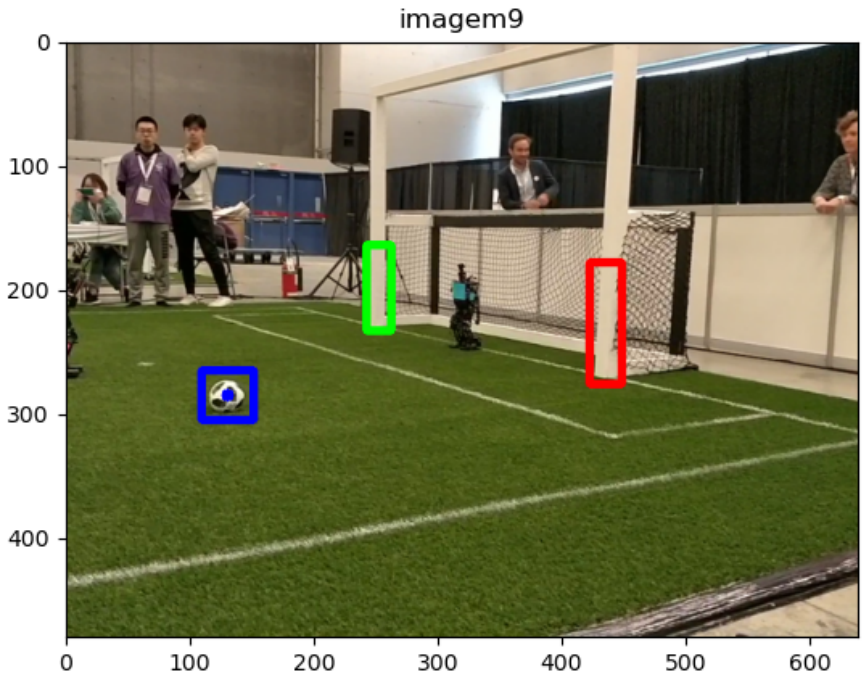

Figura 6: Detecção da Rede Neural para a imagem 9

Entre as imagens de teste, a rede neural teve resultado próximo de perfeito, tendo em vista que deseja-se identificar a bola e a parte inferior do poste. Assim sendo, essa implementação provavelmente é muito adequada para a detecção de objetos do robô humanoide.