

Relatório do Laboratório 9 - Detecção de Objetos

1 Breve Explicação em Alto Nível da Implementação

O objetivo deste relatório foi implementar uma rede neural de detecção de objetos baseada no algoritmo de YOLO (*You Only Look Once*), que, por sua vez, se baseia em uma rede neural convolucional (CNN). Em especial, a rede neural implementada terá como objetivo detectar a bola de futebol e as barras verticais do travessão (*posts*) no futebol de robôs humanoides. O código foi dividido entre a implementação da arquitetura da rede neural e o pré e pós-processamento da imagem do humanoide conforme o algoritmo YOLO. A rede neural foi implementada por meio do *framework* Keras, do Tensorflow. A saída da rede neural consiste em um vetor de *features* para cada célula do *grid* no qual a imagem foi discretizada. O pré-processamento do imagem consistiu em redimensioná-la e transformá-la em um *array* NumPy, além de normalizar os valores do canal de cores. O pós-processamento consistiu em obter a probabilidade presença da bola e dos *posts* da trave e determinar o centro do objeto, bem como os limites da *bounding box* a partir dos vetores de *features*.

1.1 Sumário do Modelo

Sumário do modelo da rede neural implementada exibido pelo método *model.summary()* do framework Keras.

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)	1,152	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)	4,608	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]

Layer (type)	Output Shape	Param #	Connected to
conv_5 (Conv2D)	(None, 30, 40, 64)	18,432	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)	36,864	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)	73,728	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)	8,192	max_pool_6[0][0]
conv_8 (Conv2D)	(None, 15, 20, 256)	294,912	leaky_relu_7[0][0]
norm_skip (BatchNormalization)	(None, 15, 20, 128)	512	conv_skip[0][0]
norm_8 (BatchNormalization)	(None, 15, 20, 256)	1,024	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)	3,850	concat[0][0]

2 Figuras Comprovando Funcionamento do Código

2.1 Detecção de Objetos com YOLO

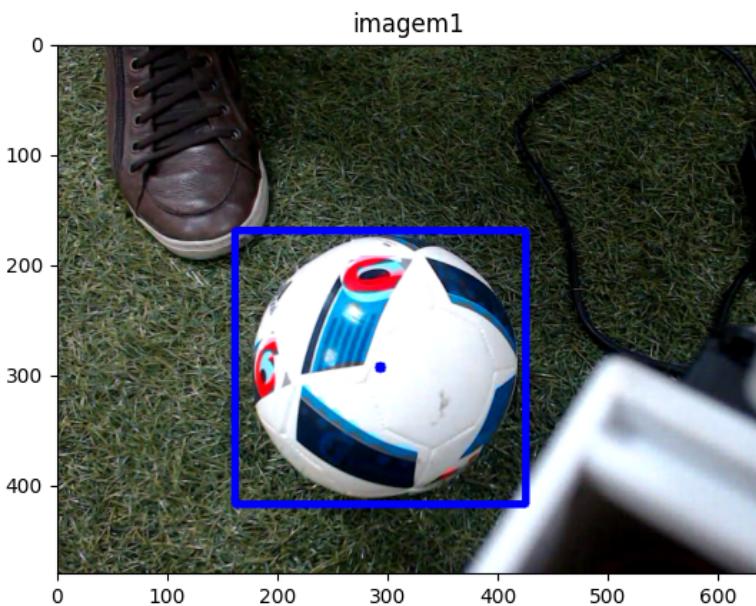


Figura 1: Exemplo 1 do teste da implementação da inferência com YOLO.

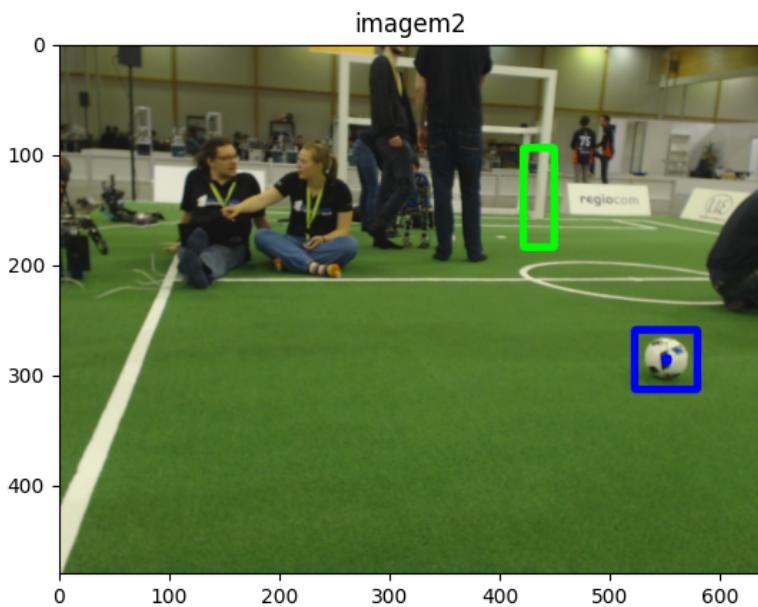


Figura 2: Exemplo 2 do teste da implementação da inferência com YOLO.

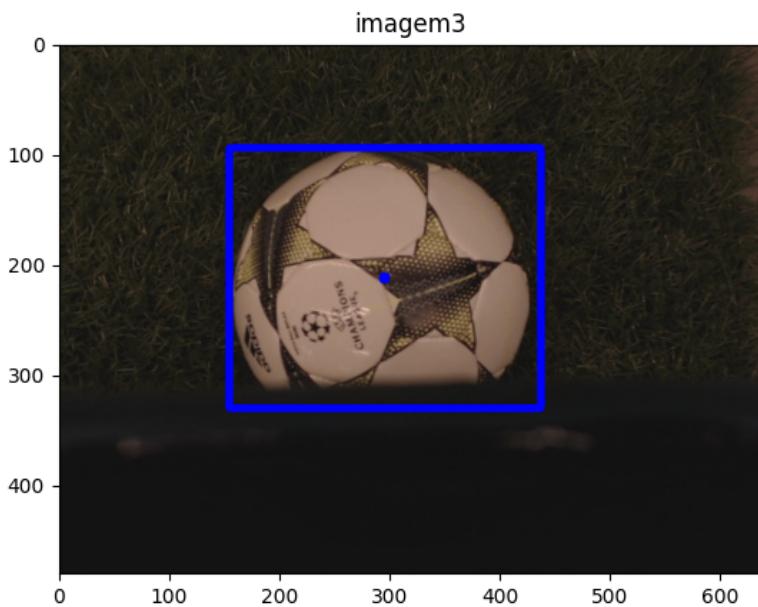


Figura 3: Exemplo 3 do teste da implementação da inferência com YOLO.

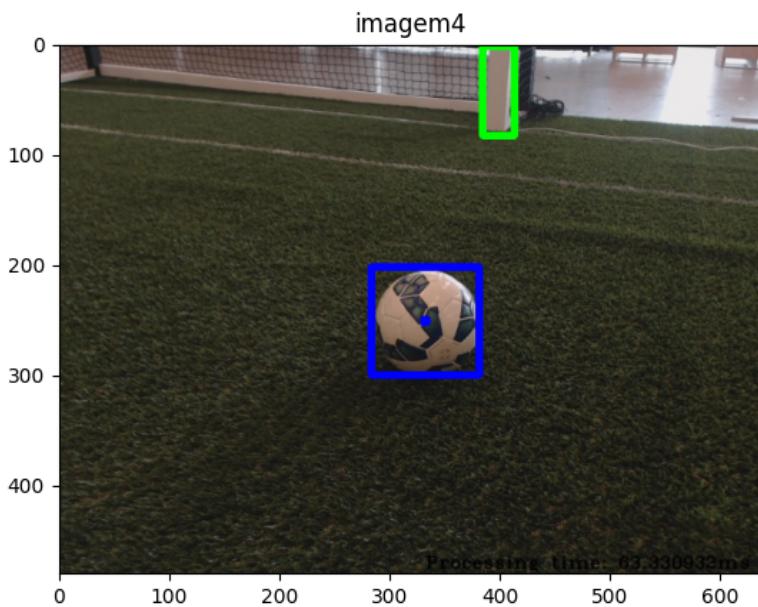


Figura 4: Exemplo 4 do teste da implementação da inferência com YOLO.

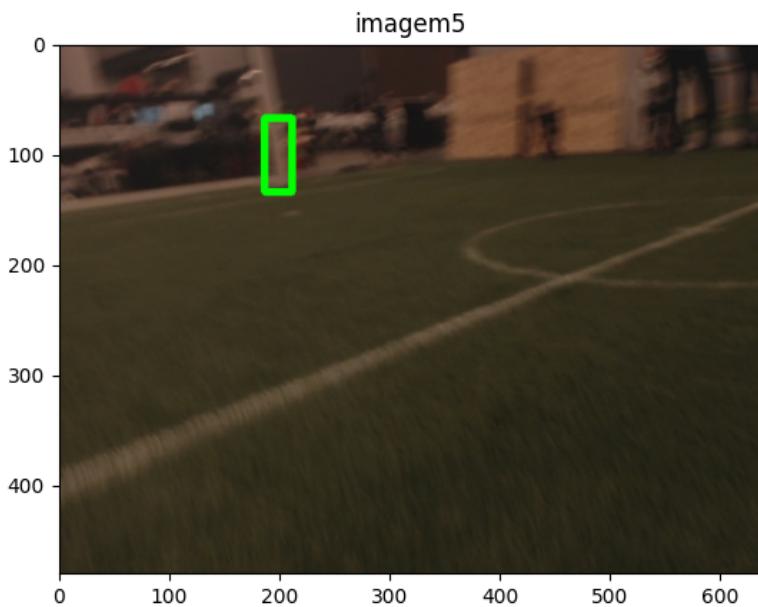


Figura 5: Exemplo 5 do teste da implementação da inferência com YOLO.

3 Discussão

A implementação da rede neural baseada no algoritmo YOLO demonstrou ser uma abordagem eficiente para a detecção de objetos no contexto do futebol de robôs humanoides. A arquitetura da rede foi cuidadosamente projetada, utilizando camadas convolucionais, normalizações e ativações não-lineares, o que contribuiu para uma boa capacidade de aprendizado e generalização. A escolha do framework Keras do TensorFlow foi estratégica, pois proporcionou um ambiente robusto e amigável para a construção e teste do modelo.

Os testes realizados evidenciam que o modelo é capaz de identificar com precisão a bola e as traves, mesmo em imagens discretizadas em grades de baixa resolução. No entanto, o desempenho pode ser influenciado por fatores como iluminação, posições complexas dos objetos e sobreposições. Esses desafios reforçam a importância do pré-processamento adequado das imagens, como a normalização dos canais de cor, e do pós-processamento, que garante a extração correta das bounding boxes.

Um aspecto a ser explorado em trabalhos futuros é a análise mais aprofundada dos hiperparâmetros, como o número de filtros e a taxa de aprendizado, para melhorar ainda mais a acurácia. Além disso, a integração de técnicas como aumentação de dados e treinamento com datasets mais diversificados pode contribuir para a robustez do modelo em cenários reais mais variados. Em geral, a implementação apresentou resultados promissores e destaca a viabilidade de soluções baseadas em redes YOLO no domínio da robótica humanoide.