Instituto Tecnológico de Aeronáutica - ITA
CT-213 - Inteligência Artificial aplicada à Robótica Móvel
Aluno: Ulisses Lopes da Silva

Relatório do Laboratório 9 - Detecção de Objetos

---

1 Breve Explicação em Alto Nível da Implementação

O presente relatório detalha a implementação de um sistema de detecção de objetos para o ambiente de futebol de robôs, cujo objetivo é identificar a bola e as traves do gol em tempo real. A solução foi desenvolvida com base em uma versão modificada do algoritmo YOLO (*You Only Look Once*), que se destaca pela sua capacidade de realizar predições em uma única passada da imagem pela rede neural, tornando-o ideal para aplicações que exigem baixa latência.

O núcleo do sistema é uma Rede Neural Convolucional (RNC) customizada, implementada com o auxílio da biblioteca Keras, sobre o framework TensorFlow. A arquitetura da rede, definida na função `make_detector_network`, foi construída de forma sequencial, empilhando camadas convolucionais (`Conv2D`), de normalização em lote (`BatchNormalization`) para estabilizar e acelerar o treinamento, funções de ativação `LeakyReLU` para introduzir não-linearidade, e camadas de `MaxPooling2D` para reduzir a dimensionalidade espacial dos mapas de features. Um aspecto notável da arquitetura é a inclusão de uma *skip connection*, implementada através de uma concatenação, que permite ao modelo combinar features de diferentes níveis de profundidade, enriquecendo a capacidade de representação da rede. A camada final da RNC foi projetada para produzir um tensor de saída com as dimensões (15, 20, 10), representando a grade de detecção e o vetor de 10 features para cada célula, conforme especificado pelo algoritmo YOLO modificado.

A lógica de detecção foi encapsulada na classe `YoloDetector`, que orquestra todo o processo. O ciclo de inferência inicia-se no método `preprocess_image`, que recebe a imagem bruta da câmera (640x480 pixels), redimensiona-a para as dimensões de entrada da rede (160x120), normaliza os valores dos pixels para o intervalo [0, 1] e a reformata para um tensor compatível. Em seguida, este tensor é passado pela rede neural, que gera o tensor de predições.

A etapa mais complexa reside no método `process_yolo_output`, responsável por decodificar o tensor de saída da RNC. A implementação percorre a grade de 15x20 células e, para cada uma, interpreta o vetor de 10 features. As probabilidades de presença de objeto (t_b e t_p) são obtidas aplicando-se a função sigmoide aos logits da rede. As coordenadas do centro do objeto e as dimensões da sua caixa delimitadora (*bounding box*) são calculadas utilizando as fórmulas do YOLO, que combinam a posição da célula na grade, os offsets preditos pela rede (também passados pela sigmoide), e as *anchor boxes* pré-definidas, escalando o resultado para o espaço de pixels da imagem original. A lógica implementada então identifica a detecção de bola com a maior probabilidade de confiança. Para as traves, o algoritmo foi projetado para encontrar as duas detecções com as maiores probabilidades, utilizando uma estrutura condicional `if/elif` robusta que garante a correta identificação dos dois melhores candidatos, mesmo que não apareçam em ordem de probabilidade durante a varredura da grade. Ao final, o sistema retorna as coordenadas

e dimensões das *bounding boxes* para o objeto "bola" e para os dois objetos "trave" com maior confiança, concluindo com sucesso o ciclo de detecção.

1.1 Sumário do Modelo

Tabela 1: Arquitetura do modelo convolucional com camadas de normalização, ativação e conexões de atalho.

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 (Batch-Normalization)	(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 (Batch-Normalization)	(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 (Batch-Normalization)	(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 (Batch-Normalization)	(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 (Batch-Normalization)	(None, 30, 40, 64)	256	conv_5[0][0]
leaky_relu_5 (LeakyReLU)	(None, 30, 40, 64)	0	norm_5[0][0]

Layer (type)	**Output Shape**	**Param #**	**Connected to**
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 (Batch-Normalization)	(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 (Batch-Normalization)	(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 (Batch-Normalization)	(None, 15, 20, 128)	512	conv_skip[0][0]
norm_8 (Batch-Normalization)	(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

2 Figuras Comprovando Funcionamento do Código

2.1 Detecção de Objetos com YOLO

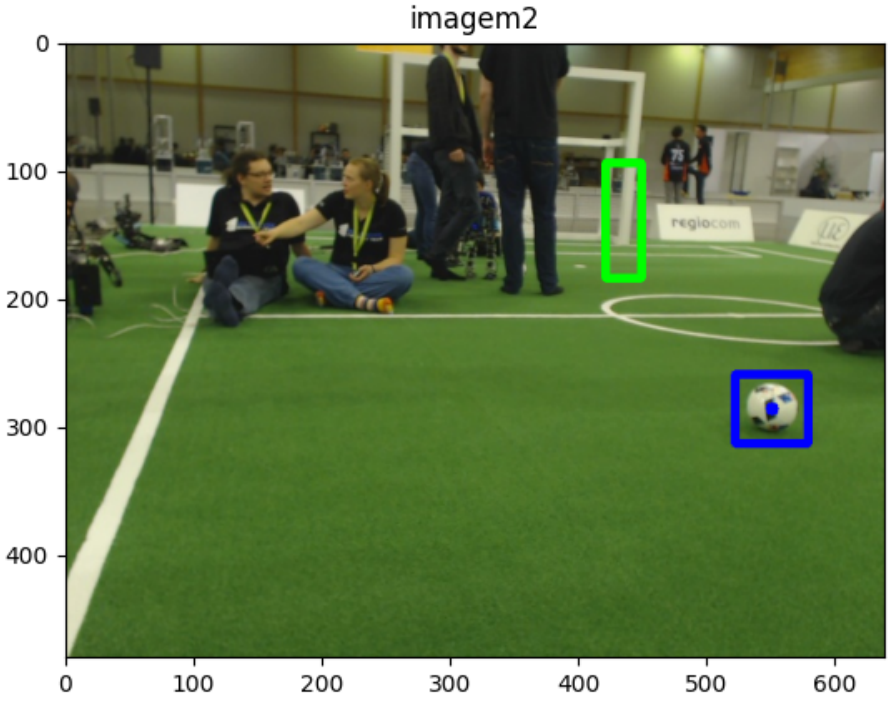

Fig. 1: Detecção da bola e trave em ambiente com presença de pessoas.

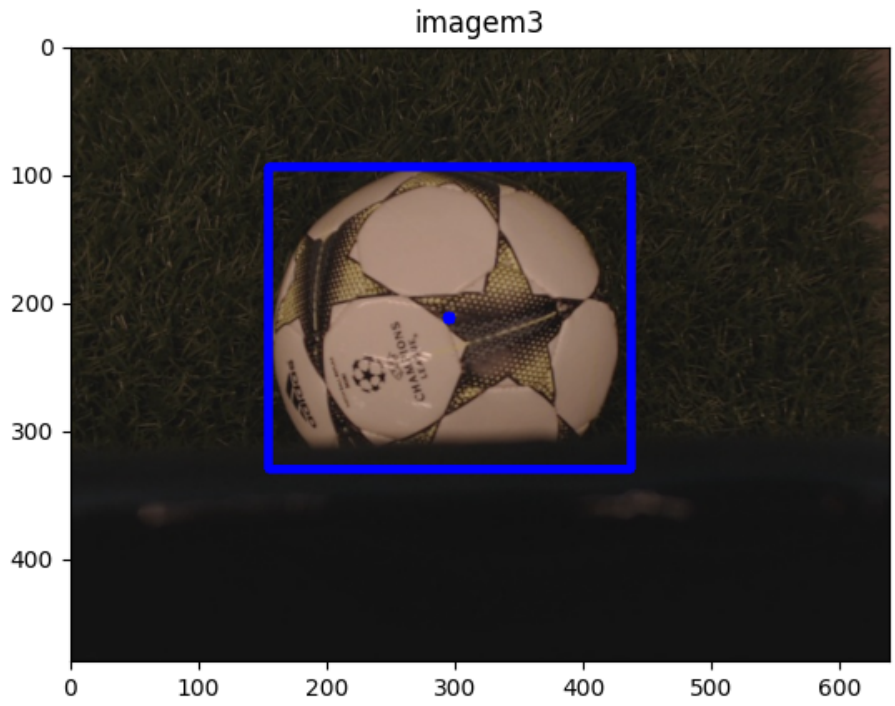

Fig. 2: Detecção da bola em destaque com fundo escuro e contraste acentuado.

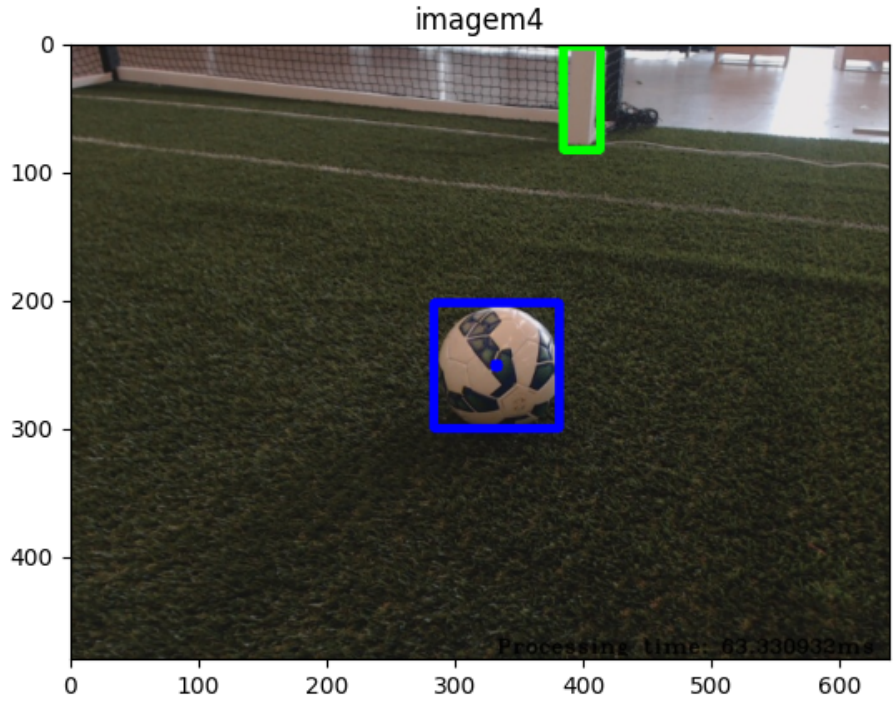

Fig. 3: Detecção da bola e da trave em ambiente iluminado e campo sintético.

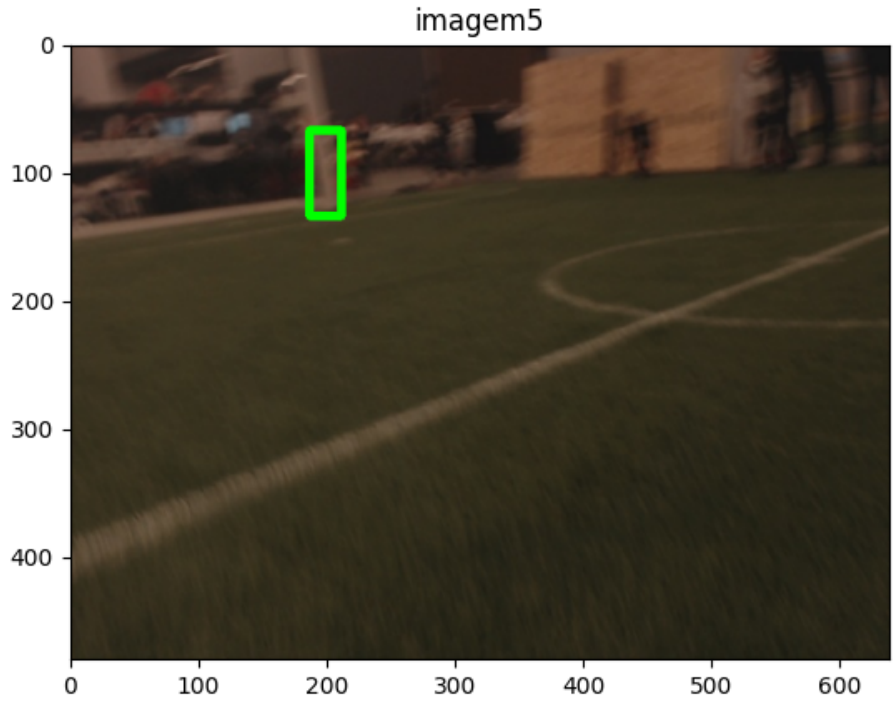

Fig. 4: Detecção apenas da trave em imagem desfocada e com baixa visibilidade.

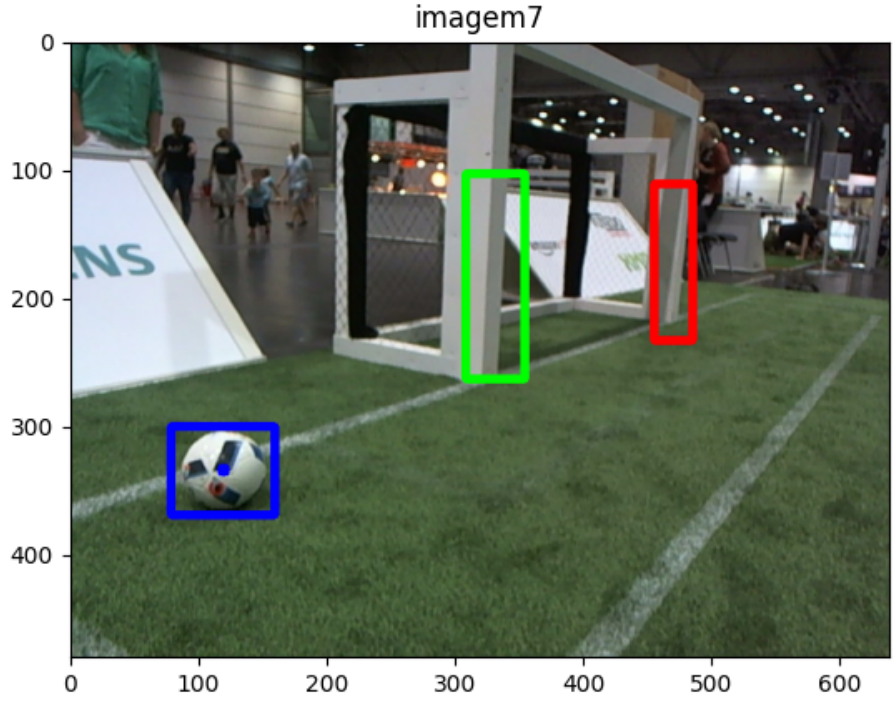

Fig. 5: Detecção da bola e das traves em uma cena com múltiplos elementos.

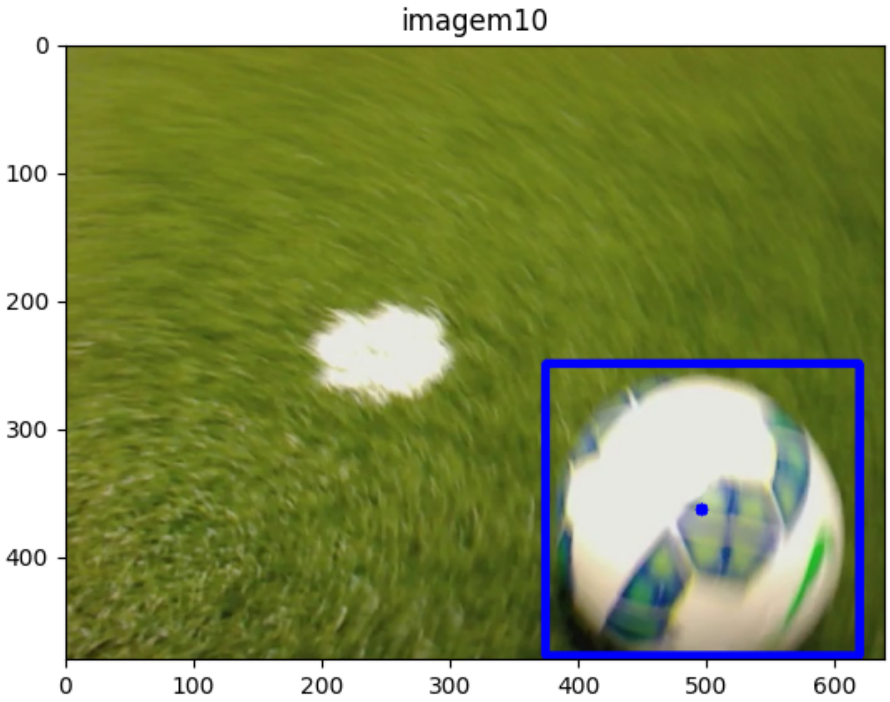

Fig. 6: Detecção precisa da bola em plano fechado, em campo com marcações.

3 Discussão

A avaliação qualitativa do algoritmo implementado foi realizada através da sua aplicação em um conjunto diversificado de imagens, que simulam diferentes condições encontradas no ambiente de competição do futebol de robôs. Os resultados, de modo geral, demonstram a alta eficácia e robustez do sistema de detecção baseado em YOLO, que se mostrou capaz de identificar corretamente a bola (delimitada por uma caixa azul) e as traves (delimitadas por caixas verdes e vermelhas) na grande maioria dos cenários.

Em situações ideais, com boa iluminação e objetos bem definidos, como observado nas imagens 2, 4 e 7, o detector apresentou um desempenho excelente. As caixas delimitadoras (*bounding boxes*) foram posicionadas com notável precisão, envolvendo firmemente os objetos de interesse. A imagem 7 é um exemplo particularmente interessante, pois demonstra o sucesso da lógica implementada para a detecção das duas traves com maior probabilidade. A identificação de uma trave com a caixa verde (primeiro candidato) e a outra com a caixa vermelha (segundo candidato) valida a correta implementação do algoritmo de seleção para múltiplos objetos de uma mesma classe. Adicionalmente, em cenários onde um dos objetos não estava presente, como a ausência de traves na imagem 3, o modelo corretamente não gerou falsos positivos, demonstrando boa capacidade de generalização.

É particularmente interessante notar o comportamento do detector em condições adversas, que são comuns em um ambiente robótico dinâmico. As imagens 5 e 10, por exemplo, apresentam um forte efeito de *motion blur*, causado pelo movimento rápido da câmera. Mesmo com a deformação e a perda de nitidez dos objetos, o algoritmo foi capaz de localizar com sucesso a trave na imagem 5 e a bola na imagem 10. Esse desempenho frente ao borramento de movimento é um indicativo crucial da viabilidade do modelo para aplicação em um robô real, que está constantemente em

movimento.

Em suma, a análise dos resultados visuais confirma que a arquitetura da Rede Neural Convolucional e a lógica de pós-processamento baseada em YOLO foram implementadas com sucesso, bem como demonstra um desempenho bastante satisfatório. O sistema não só performa com alta precisão em cenários controlados, mas também demonstra uma resiliência fundamental a desafios práticos como variações de iluminação, oclusão parcial, complexidade do plano de fundo e, mais importante, ao *motion blur*, validando sua aplicação para a tarefa de detecção de objetos no futebol de robôs, configurando o algoritmo YOLO como uma excelente ferramenta para essa tarefa.