Inteligência Artificial para Robótica Móvel CT-213

Instituto Tecnológico de Aeronáutica

Relatório do Laboratório 9 - Detecção de Objetos

Leonardo Peres Dias

1 de junho de 2025

Sumário

1 Breve Explicação em Alto Nível da Implementação

A rede neural convolucional implementada para detecção de objetos segue uma arquitetura inspirada no modelo YOLO (You Only Look Once), adaptada para imagens com resolução de 120×160 pixels. A saída da rede possui dimensões $(15, 20, 10)$, correspondendo a uma grade de 15×20 células, cada uma com um vetor de 10 atributos.

A arquitetura é composta por nove camadas convolucionais principais, intercaladas com *Batch Normalization* e funções de ativação Leaky ReLU. O modelo pode ser dividido em três blocos principais:

- **Extração de características**: As primeiras camadas convolucionais extraem características locais da imagem, com aumento progressivo da profundidade do mapa de ativação (de 8 até 64 filtros), seguidas por operações de *MaxPooling* que reduzem a resolução espacial.
- **Bloco de salto (skip connection)**: Após a sexta camada convolucional, é realizada uma conexão residual, permitindo o reaproveitamento de características de nível intermediário.
- **Concatenação e previsão**: As saídas da skip connection e da oitava camada convolucional são concatenadas. Em seguida, uma última convolução 1×1 com 10 filtros gera a saída final, representando as probabilidades e parâmetros das bounding boxes para bola e traves.

Já a classe `YoloDetector` implementa a lógica de detecção de objetos (bola e traves) a partir de imagens da câmera do robô, utilizando uma rede neural do tipo YOLO treinada previamente e salva no formato `.hdf5`.

- **Inicialização do modelo**: no método `__init__()`, o modelo é carregado com a função `load_model` do Keras e os tamanhos das *anchor boxes* para bola e trave são definidos.
- **Pré-processamento da imagem**: o método `preprocess_image()` redimensiona a imagem de entrada de 640×480 para 160×120 e normaliza os pixels para o intervalo $[0, 1]$. O formato final da imagem se torna $(1, 120, 160, 3)$, compatível com a entrada da rede.
- **Inferência e detecção**: o método `detect()` realiza a predição da rede neural com a imagem processada e chama `process_yolo_output()` para extrair as detecções.

- **Processamento da saída da rede**: o método `process_yolo_output()` interpreta a saída da rede YOLO, que possui formato $(15, 20, 10)$. Cada célula da grade representa uma sub-região da imagem e contém:

  – Para a bola: $(t_b, t_{xb}, t_{yb}, t_{wb}, t_{hb})$

  – Para as traves: $(t_p, t_{xp}, t_{yp}, t_{wp}, t_{hp})$

  Os valores t são transformados, sendo posteriormente multiplicados por fatores de escala e pelas dimensões das *anchor boxes*. A célula com maior probabilidade para a bola é selecionada, e as duas melhores para as traves são escolhidas.

- **Formato final da detecção**: cada objeto detectado é representado por uma tupla de 5 elementos: (probabilidade, x, y, largura, altura), indicando a confiança da detecção e a caixa delimitadora no espaço da imagem original.

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 (BatchNormalizatio…	(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 (BatchNormalizatio…	(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 (BatchNormalizatio…	(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 (BatchNormalizatio…	(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)	18,432	max_pool_4[0][0]
norm_5 (BatchNormalizatio…	(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 (BatchNormalizatio…	(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 (BatchNormalizatio…	(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 (BatchNormalizatio…	(None, 15, 20, 128)	512	conv_skip[0][0]
norm_8 (BatchNormalizatio…	(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[… leaky_relu_8[0][…
conv_9 (Conv2D)	(None, 15, 20, 10)	3,850	concat[0][0]

Total params: 445,346 (1.70 MB)
Trainable params: 443,938 (1.69 MB)
Non-trainable params: 1,408 (5.50 KB)

2 Figuras Comprovando Funcionamento do Código

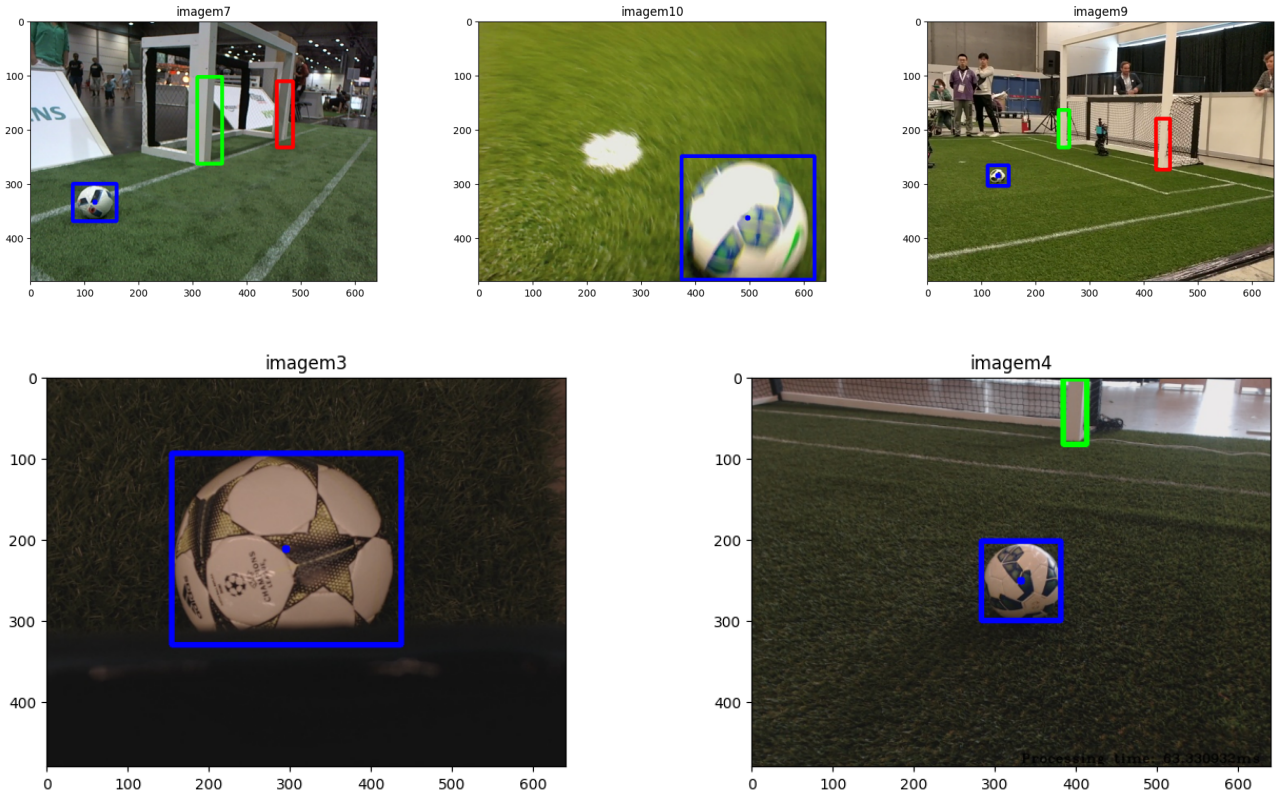

Figura 1: Exemplos de deteção feitas pela YOLO.

3 Discussão

As figuras mostram que o detector é capaz de localizar corretamente a bola (caixa azul) e as traves (caixas verde e vermelha) em diferentes cenários. A bola foi detectada com precisão em diversas escalas e posições, mesmo quando parcialmente ocluída. As traves também foram bem identificadas, inclusive em imagens com perspectiva ou múltiplos elementos no fundo. Os resultados indicam que a rede generaliza bem para diferentes situações do jogo.