Detecção de pequenos objetos com YOLO e DOTA

ESTUDO COMPARATIVO

Inatel - Instituto Nacional de Telecomunicações
TP557 - Tópicos Avançados em IoT e Machine Learning
Jonas V. Moreira, Jul - 2025

Objetivo

Comparar diferentes versões do modelo YOLO para detecção de pequenos objetos utilizando o dataset DOTA.

- Avaliar diferentes versões do algoritmo YOLO aplicadas à detecção de pequenos objetos em imagens aéreas
- Identificar o modelo com melhor desempenho para aplicações em TinyML
- Implantar o modelo otimizado em um dispositivo embarcado (Raspberry Pi)

Contexto

- **Desafios na detecção de pequenos objetos**
  Imagens de alta resolução frequentemente apresentam objetos com dimensões reduzidas, dificultando a identificação devido à baixa resolução local, forte presença de ruído e sobreposição com o cenário de fundo.

- **Principais aplicações**
  Soluções em vigilância inteligente, monitoramento ambiental automatizado e segurança pública, onde a identificação precisa de objetos em escala reduzida é essencial para tomada de decisões rápidas e confiáveis.

- **Base de dados utilizada:**
  DOTA (Dataset for Object Detection in Aerial Images) - Conjunto especializado em imagens aéreas com anotações orientadas (OBB), abrangendo diversas categorias e escalas, capturadas por sensores embarcados em drones, satélites e aeronaves.

Yolo

- O YOLO (You Only Look Once) é um algoritmo de detecção de objetos em tempo real, capaz de predizer a localização e a classe de um objeto em uma imagem, sendo amplamente utilizado na área de visão computacional

Dota dataset

- O DOTA é um conjunto de dados que dá ênfase à deteção de objectos em imagens aéreas com Oriented Bounding Boxes (OBB).
  - v1
    - Aproximadamente 2.800 imagens
    - Mais de 188.000 instâncias
    - 15 categorias

Modelos Candidatos

- Yolov5
- Yolov8
- Yolov11
- Yolov12

Versões utilizadas

- N e M

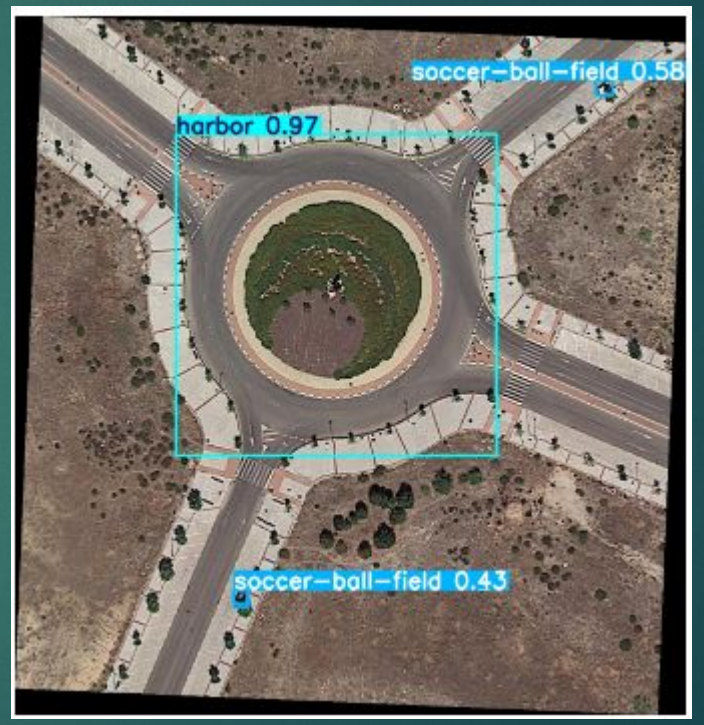
Desenvolvimento

- Metodologia: treinar todos os modelos com as mesmas configurações de hiperparâmetros e dataset.
- Dota dataset
  - V1 (aproximadamente 2.800 imagens, com 188.000 instâncias, em 15 categorias)
- Fase 1 – Conversão dos Labels de OBB para HBB
- Fase 2 – Treinamento local (Anaconda)
- Fase 3 – Treinamento na nuvem (Colab Pro)
- Fase 4 – Refinamento do Modelo
  - Ajuste dos hiperparâmetros
  - Treinamento com dataset fatiado
- Fase 5 – Dispositivo IoT
  - Exportando o modelo para o formato NCNN
  - Embarcando o modelo para o Raspberry Pi5

Fase 1 - Conversão dos labels

Diferença entre OBB e HBB

Oriented Bounding Box (OBB)



Horizontal Bounding Box (HBB)

Fase 2 - Treinamento local

Resultados

TABLE I
PERFORMANCE COMPARISON OF YOLO MODELS TRAINED WITH CPU

Model	mAP @ 0.50	mAP @ 0.50:0.95	Precision	Recall	Latency (CPU)	Loss Final (train)	Loss Final (val)	Size
yolov5n	0.394	0.207	0.651	0.378	90 ms	0.244	0.246	13.6 MB
yolov5m	0.496	0.319	0.723	0.455	268 ms	2.488	2.837	48.1 MB
yolov8n	0.334	0.199	0.629	0.314	37 ms	3.414	3.747	6.2 MB
yolov8m	0.437	0.270	0.612	0.411	273 ms	3.290	3.791	52 MB
yolov11n	0.392	0.240	0.704	0.359	55 ms	3.139	3.511	5.4 MB
yolov11m	0.445	0.276	0.637	0.412	285 ms	3.321	3.672	40.5 MB
yolov12n	0.176	0.091	0.564	0.183	64 ms	5.243	5.454	5.5 MB
yolov12m	0.441	0.272	0.645	0.401	332 ms	3.403	3.784	40.7 MB

Fase 3 - Treinamento no Colab Pro

Treinamento no Google Colab Pro

TABLE II
PERFORMANCE COMPARISON OF YOLO MODELS TRAINED WITH GPU (GOOGLE COLAB)

Model	mAP @ 0.50	mAP @ 0.50:0.95	Precision	Recall	Latency (CPU)	Loss Final (train)	Loss Final (val)	Size
yolov5n	0.402	0.248	0.682	0.374	87 ms	3.024	3.395	5 MB
yolov5m	0.516	0.340	0.762	0.465	265 ms	2.230	2.870	48.1 MB
yolov8n	0.416	0.257	0.665	0.385	48 ms	2.887	3.333	6.2 MB
yolov8m	0.517	0.339	0.762	0.467	270 ms	2.146	2.872	52 MB
yolov11n	0.424	0.264	0.711	0.387	54 ms	2.865	3.294	5.5 MB
yolov11m	0.543	0.358	0.724	0.506	290 ms	2.161	2.786	40.5 MB
yolov12n	0.428	0.268	0.661	0.400	63 ms	2.794	3.243	5.5 MB
yolov12m	0.531	0.350	0.732	0.487	338 ms	2.193	2.790	40.8 MB

Fase 4 - Refinando o Modelo

Treinamento no Google Colab Pro, com ajuste de hiperparâmetros

TABLE III
PERFORMANCE COMPARISON OF YOLO MODELS TRAINED WITH GPU (GOOGLE COLAB) — EXTENDED ANALYSIS

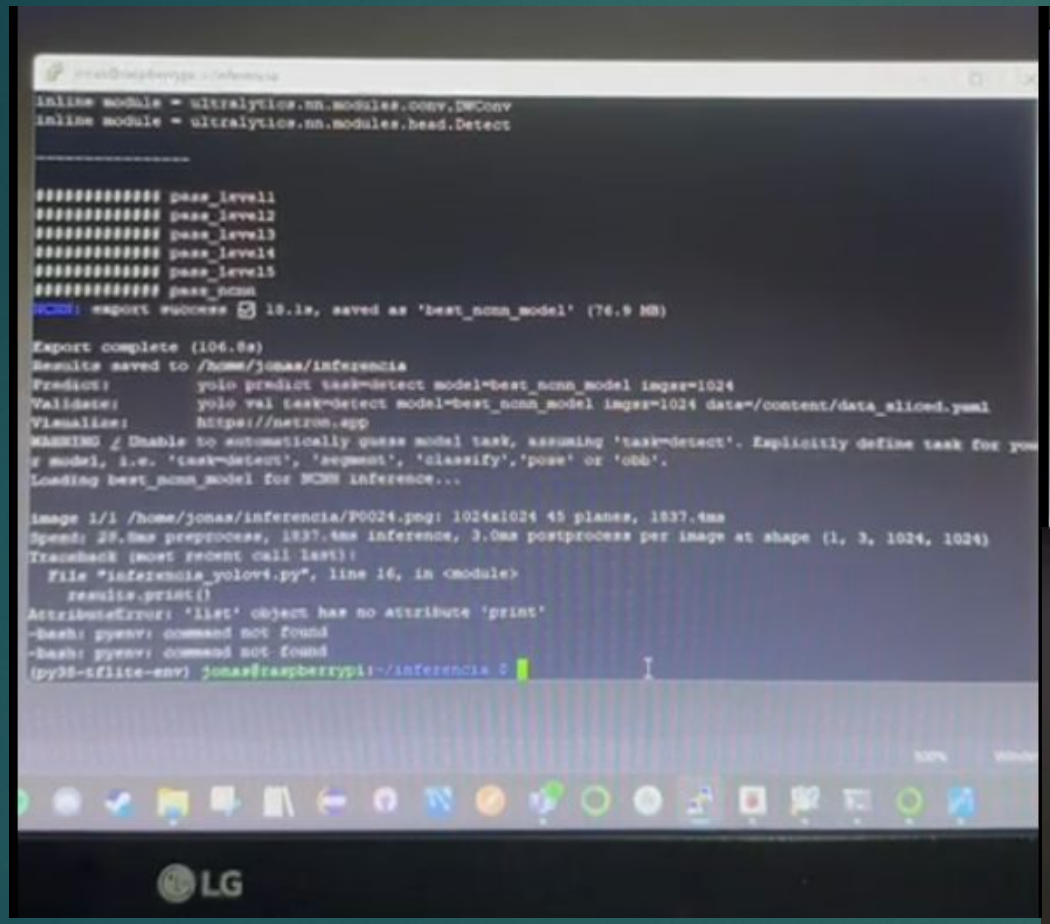
Model	mAP @ 0.50	mAP @ 0.50:0.95	Precision	Recall	Latency (CPU)	Loss Final (train)	Loss Final (val)	Size
yolov5m	0.593	0.397	0.766	0.551	613 ms	2.238	2.760	50.5 MB
yolov8m	0.596	0.403	0.774	0.553	736 ms	2.134	2.755	52.1 MB
yolov11m	0.608	0.415	0.772	0.564	813 ms	2.204	2.704	40.5 MB

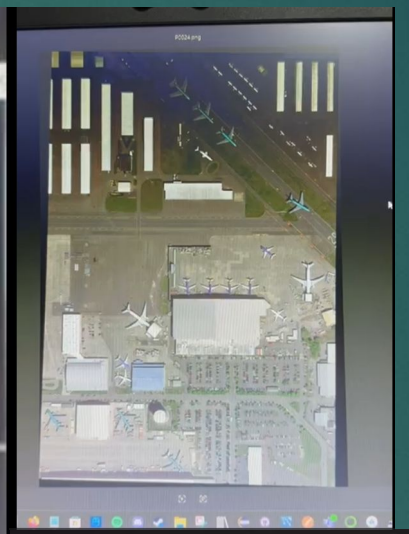
Treinamento no Google Colab Pro, com dataset de imagens fatiadas

TABLE IV
PERFORMANCE COMPARISON OF THE YOLOV11M MODEL UNDER DIFFERENT EVALUATION CONDITIONS

Model	mAP @ 0.50	mAP @ 0.50:0.95	Precision	Recall	Latency (CPU)	Loss Final (train)	Loss Final (val)	Size
yolov11m (OD)	0.543	0.372	0.721	0.513	807 ms	2.322	3.087	40.5 MB
yolov11m (SD)	0.738	0.504	0.768	0.700	818 ms	2.322	3.087	40.5 MB

Fase 5 - TinyML





Conclusão

- Pré-processamento e engenharia de dados foram cruciais
- YOLOv11m apresentou bom desempenho para aplicações embarcadas

- Trabalhos Futuros
  - Explorar novos hiperparâmetros
  - Aplicar técnicas de processamento nas imagens de teste
  - Comparar com outras arquiteturas

Fim

Perguntas?