UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Guilherme dos Santos Silva

Análise Comparativa de Algoritmos de Detecção de Objetos para Inspeção de Trens de Pouso

Uberlândia, Brasil

2026

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Guilherme dos Santos Silva

Análise Comparativa de Algoritmos de Detecção de Objetos para Inspeção de Trens de Pouso

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Henrique Coelho Fernandes

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2026

Guilherme dos Santos Silva

Análise Comparativa de Algoritmos de Detecção de Objetos para Inspeção de Trens de Pouso

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Sistemas de Informação.

Trabalho aprovado. Uberlândia, Brasil, 28 de maio de 2026:

Prof. Dr. Henrique Coelho Fernandes
Orientador

Me. Monhel Maudoony Pierre
Examinador

Prof. Dr. Daniel Duarte Abdala
Examinador

Uberlândia, Brasil
2026

Agradecimentos

Aos meus pais, Marcia e Julio, pelo amor incondicional, pela confiança depositada em cada escolha que fiz e pelo incentivo constante aos estudos. Sem o suporte que vocês sempre ofereceram, nenhuma desta e de outras conquistas teria sido possível.

À minha namorada, Geovana, pela presença e pelo companheirismo ao longo desses anos de graduação. Obrigado por compreender as ausências, por torcer genuinamente e por tornar essa jornada mais leve.

Ao Prof. Dr. Henrique Coelho Fernandes, pela orientação cuidadosa e pela disponibilidade ao longo de todo o desenvolvimento deste trabalho.

À minha família e aos meus amigos, pela presença e pelo incentivo de sempre. O apoio de cada um de vocês foi parte essencial dessa trajetória.

À Universidade Federal de Uberlândia, minha gratidão pelas oportunidades de aprendizado e crescimento que me proporcionou, dentro e fora da sala de aula.

Resumo

A inspeção visual do trem de pouso de aeronaves é uma etapa crítica dos processos de Manutenção, Reparo e Operações (MRO), realizada tradicionalmente de forma manual e sujeita a inconsistências humanas. Este trabalho propõe e avalia comparativamente dois paradigmas arquiteturais de detecção de objetos para a automatização dessa tarefa: o **YOLO11s**, baseado em Redes Neurais Convolucionais (CNN), e o **RT-DETR-L**, baseado em *Vision Transformers*. Ambos os modelos foram treinados sob condições experimentais rigorosamente equalizadas — 50 épocas, mesma plataforma de *hardware* (GPU NVIDIA Tesla T4), mesmo *dataset* (*Landing-Gear IRP*, 2.979 imagens com *data augmentation*) e mesmos hiperparâmetros padrão — utilizando *Transfer Learning* a partir de pesos pré-treinados no COCO. Os resultados demonstraram a viabilidade técnica de ambos os paradigmas, com desempenhos superiores aos reportados em trabalhos relacionados: o YOLO11s alcançou mAP@50 de 79,03% (precisão: 81,35%, recall: 75,18%) e velocidade de inferência de 62,5 FPS, enquanto o RT-DETR-L registrou mAP@50 de 75,94% com recall superior de 78,50%, F1 máximo de 0,80 e 18,7 FPS. A análise comparativa revelou *trade-offs* bem definidos: o YOLO11s apresenta maior precisão de localização (mAP@50-95: 37,76% vs 34,04%) e opera acima do limiar de 30 FPS necessário para vídeo em tempo real, sendo recomendado para sistemas autônomos com restrições de *hardware*; o RT-DETR-L reduziu a taxa de falsos negativos de 17,3% para 12,0% (30 instâncias adicionais detectadas), sendo preferível em sistemas de assistência à inspeção com validação humana, onde a minimização de componentes não detectados é prioritária.

Palavras-chave: Visão Computacional, Detecção de Objetos, YOLO, RT-DETR, Inspeção Aeronáutica, Trem de Pouso.

Lista de ilustrações

Figura 1 – Modelo não linear de um neurônio artificial. Adaptado de Haykin (2009). 15
Figura 2 – Arquitetura clássica da LeNet-5, um marco na evolução das CNNs. Adaptado de (LECUN et al., 1998). 16
Figura 3 – Visão geral da arquitetura YOLO11, mostrando os componentes principais: Backbone, Neck e Head. Fonte: (KHANAM; HUSSAIN, 2024). . 21
Figura 4 – Visão geral da arquitetura RT-DETR, mostrando os componentes principais: Backbone, Efficient Hybrid Encoder (AIFI e CCFM), IoU-aware Query Selection e Decoder & Head. Fonte: (ZHAO et al., 2023). 25
Figura 5 – Curvas de treinamento e validação do YOLO11s ao longo das 50 épocas. Os gráficos superiores mostram a evolução das funções de perda (*box loss*, *cls loss*, *dfl loss*) para o conjunto de treinamento, enquanto os gráficos inferiores exibem as mesmas métricas para validação. Os dois últimos gráficos à direita apresentam a evolução da precisão, recall, mAP@50 e mAP@50-95. 39
Figura 6 – Curva Precisão-Revocação (PR Curve) do YOLO11s para a classe *landing_gear*. A área sob a curva (AUC) representa o valor de mAP@50 = 0,790. 41
Figura 7 – Curvas de F1-Confidence, Precision-Confidence e Recall-Confidence do YOLO11s. Estas curvas auxiliam na determinação do limiar de confiança ótimo para inferência. 42
Figura 8 – Matriz de confusão absoluta e normalizada do YOLO11s avaliada no conjunto de validação. 44
Figura 9 – Curvas de treinamento e validação do RT-DETR-L ao longo de 50 épocas. Os gráficos mostram a evolução das funções de perda (GIoU, classificação, L1) e das métricas de desempenho (Precision, Recall, mAP@50, mAP@50-95). 46
Figura 10 – Curva Precisão-Revocação (PR Curve) do RT-DETR-L para a classe *landing_gear*. A área sob a curva (AUC) representa o valor de mAP@50 = 0,7594. 48
Figura 11 – Curvas de F1-Confidence, Precision-Confidence e Recall-Confidence do RT-DETR-L. Estas curvas revelam diferenças comportamentais importantes em relação ao YOLO11s e auxiliam na determinação do limiar de confiança ótimo para inferência. 49
Figura 12 – Matriz de confusão absoluta e normalizada do RT-DETR-L avaliada no conjunto de validação. 51

Figura 13 – Anotações de referência (*ground truth*) do lote de validação. As caixas delimitadoras indicam todas as instâncias de *landing_gear* anotadas manualmente no dataset. 53
Figura 14 – Predições do YOLO11s no lote de validação. Cada caixa delimitadora é acompanhada do *score* de confiança da detecção. 54
Figura 15 – Predições do RT-DETR-L no mesmo lote de validação. Observar os *scores* de confiança geralmente mais elevados em comparação ao YOLO11s, reflexo do limiar ótimo mais alto (0,603 vs 0,450). 55

Lista de tabelas

Tabela 1 – Principais hiperparâmetros de treinamento determinados automaticamente pelo framework Ultralytics. 36
Tabela 2 – Comparação de desempenho entre YOLO11s e RT-DETR-L no dataset Landing-Gear IRP (50 épocas). 56

Lista de abreviaturas e siglas

AOG	Aircraft on Ground (Aeronave em Solo)
AP	Average Precision (Precisão Média)
CCFM	Cross-scale Feature Mixing with Fusion
CNN	Convolutional Neural Network (Rede Neural Convolucional)
DETR	DEtection TRansformer
FN	Falso Negativo (False Negative)
FP	Falso Positivo (False Positive)
FPN	Feature Pyramid Network (Rede de Pirâmide de Características)
FPS	Frames Per Second (Quadros por Segundo)
GPU	Graphics Processing Unit (Unidade de Processamento Gráfico)
IoU	Intersection over Union (Interseção sobre União)
mAP	mean Average Precision (Média de Precisão Média)
ML	Machine Learning (Aprendizado de Máquina)
MRO	Manutenção, Reparo e Operações
NMS	Non-Maximum Suppression
PAN	Path Aggregation Network
RNA	Rede Neural Artificial
RT-DETR	Real-Time DEtection TRansformer
TCC	Trabalho de Conclusão de Curso
TP	Verdadeiro Positivo (True Positive)
UAV	Unmanned Aerial Vehicle (Veículo Aéreo Não Tripulado)
ViT	Vision Transformer
YOLO	You Only Look Once

Sumário

1 INTRODUÇÃO 12

2 FUNDAMENTAÇÃO TEÓRICA 14
2.1 Aprendizado de Máquina (Machine Learning) 14
2.1.1 Conceitos Fundamentais 14
2.1.2 Aprendizado Supervisionado 14
2.2 Redes Neurais Artificiais e Deep Learning 15
2.2.1 O Neurônio Artificial 15
2.2.2 Deep Learning e Redes Multicamadas 15
2.3 Redes Neurais Convolucionais (CNNs) 16
2.3.1 Arquitetura e Camadas Principais 17
2.3.2 Aprendizagem por Transferência (Transfer Learning) 17
2.4 Visão Computacional 18
2.5 Detecção de Objetos 19
2.6 Arquiteturas para Detecção de Objetos 20
2.6.1 Detectores de Estágio Único: A Família YOLO 20
2.6.1.1 Backbone: Extração Hierárquica de Características 21
2.6.1.2 Neck: Fusão Multi-Escala de Características 22
2.6.1.3 Head: Predição Multi-Escala 22
2.6.1.4 Eficiência Computacional e Escalabilidade 23
2.6.2 Vision Transformers (ViT) e Mecanismos de Atenção 24
2.6.2.1 Do Paradigma Convolucional aos Transformers 24
2.6.2.2 Arquitetura do RT-DETR: Visão Geral 24
2.6.2.3 Backbone: Extração de Características Multi-Escala 25
2.6.2.4 Efficient Hybrid Encoder: Fusão Inteligente de Características 25
2.6.2.4.1 AIFI: Attention-based Intrascale Feature Interaction 25
2.6.2.4.2 CCFM: Cross-scale Feature Mixing with Fusion 26
2.6.2.5 IoU-aware Query Selection: Inicialização Inteligente 26
2.6.2.6 Decoder & Head: Predição via Atenção Cruzada 27
2.6.2.7 Vantagens e Compromissos do Paradigma Transformer 27
2.7 Métricas de Avaliação de Desempenho 28
2.7.1 Intersection over Union (IoU) 28
2.7.2 Precisão, Revocação e F1-Score 28
2.7.3 Mean Average Precision (mAP) 29

3 TRABALHOS RELACIONADOS 31

4 METODOLOGIA PROPOSTA . . . 33
4.1 Ambiente de Desenvolvimento e Hardware . . . 33
4.2 Base de Dados (*Dataset*) . . . 33
4.2.1 Pré-processamento e Aumento de Dados (*Data Augmentation*) . . . 34
4.2.2 Distribuição do Conjunto de Dados . . . 35
4.3 Implementação e Execução do Treinamento . . . 35
4.3.1 Seleção dos Modelos . . . 35
4.3.2 Configuração do Treinamento . . . 36
4.3.3 Procedimento de Treinamento e Persistência de Dados . . . 36
4.3.4 Refinamento Iterativo . . . 37

5 RESULTADOS E DISCUSSÃO . . . 38
5.1 Configuração Experimental . . . 38
5.1.0.0.1 Tempo de Treinamento: . . . 39
5.2 Resultados do YOLO11s . . . 39
5.2.1 Curvas de Treinamento . . . 39
5.2.1.0.1 Perdas de Treinamento (YOLO11s): . . . 39
5.2.1.0.2 Perdas de Validação (YOLO11s): . . . 40
5.2.1.0.3 Métricas de Desempenho (YOLO11s): . . . 40
5.2.2 Curvas de Avaliação . . . 41
5.2.2.1 Curva Precisão-Revocação . . . 41
5.2.2.2 Curvas de Confiança . . . 42
5.2.2.2.1 F1-Confidence Curve: . . . 43
5.2.2.2.2 Precision-Confidence Curve: . . . 43
5.2.2.2.3 Recall-Confidence Curve: . . . 43
5.2.3 Matriz de Confusão e Análise de Erros . . . 44
5.2.3.0.1 Verdadeiros Positivos (TP): . . . 45
5.2.3.0.2 Falsos Negativos (FN): . . . 45
5.2.3.0.3 Análise Normalizada: . . . 45
5.3 Resultados do RT-DETR-L . . . 45
5.3.1 Curvas de Treinamento . . . 45
5.3.1.0.1 GIoU Loss (Generalized Intersection over Union): . . . 46
5.3.1.0.2 Classification Loss: . . . 46
5.3.1.0.3 L1 Loss: . . . 46
5.3.1.0.4 Perdas de Validação (RT-DETR-L): . . . 47
5.3.1.0.5 Métricas de Desempenho (RT-DETR-L): . . . 47
5.3.2 Curvas de Avaliação . . . 48
5.3.2.1 Curva Precisão-Revocação . . . 48
5.3.2.2 Curvas de Confiança . . . 49
5.3.2.2.1 F1-Confidence Curve: . . . 50

5.3.2.2.2 Precision-Confidence Curve: . . . 50
5.3.2.2.3 Recall-Confidence Curve: . . . 50
5.3.3 Matriz de Confusão e Análise de Erros . . . 51
5.3.3.0.1 Verdadeiros Positivos (TP): . . . 52
5.3.3.0.2 Falsos Negativos (FN): . . . 52
5.3.3.0.3 Falsos Positivos (FP): . . . 52
5.3.3.0.4 Análise Normalizada: . . . 52
5.4 Avaliação Qualitativa: Análise Visual das Predições . . . 52
5.4.1 Casos de Sucesso . . . 53
5.4.2 Padrões de Erro Observados . . . 55
5.5 Análise Comparativa e Discussão . . . 56
5.5.1 Comparação Quantitativa de Desempenho . . . 56
5.5.2 Principais Achados e Trade-offs entre Paradigmas . . . 57
5.5.2.0.1 mAP e Precisão de Localização: . . . 57
5.5.2.0.2 Recall e Cobertura de Detecção: . . . 57
5.5.2.0.3 Precisão individual das detecções e perfil de confiança: . . . 57
5.5.2.0.4 Complexidade Computacional: . . . 57
5.5.2.0.5 Contextualização com a Literatura: . . . 58
5.5.3 Implicações Práticas para MRO Aeronáutico . . . 58
5.5.3.0.1 Sistema de Assistência à Inspeção com validação humana — Recomendação: RT-DETR-L: 58
5.5.3.0.2 Sistema Totalmente Autônomo ou com Restrições de Hardware — Recomendação: YOLO11s: . . . 58
5.5.4 Limitações e Trabalhos Futuros . . . 59
5.5.4.0.1 Treinamento Estendido do RT-DETR: . . . 59
5.5.4.0.2 Avaliação de Velocidade de Inferência: . . . 59
5.5.4.0.3 Detecção Multi-classe e Robustez de Treinamento: . . . 59
5.5.4.0.4 Análise de Escalabilidade e Viabilidade Econômica: . . . 60
6 CONCLUSÃO . . . 61
REFERÊNCIAS . . . 63

1 Introdução

A indústria da aviação, um pilar da economia e sociedade modernas, estabelece sua segurança operacional em processos de Manutenção, Reparo e Operações (MRO) de extrema rigidez. Dentre estes processos, a inspeção visual de componentes críticos, como o trem de pouso, constitui uma tarefa recorrente e fundamental. Embora estabelecido, esse processo é tradicionalmente realizado de forma manual por técnicos especializados, tornando-o intensivo em tempo, custo e suscetível a inconsistências que podem variar entre inspetores (MEKER et al., 2024). Diante deste cenário, e considerando os desafios e as oportunidades de otimização na indústria de MRO, a automação por meio de sistemas de visão computacional surge como uma área de pesquisa e desenvolvimento promissora para aumentar a eficiência, a rastreabilidade e a confiabilidade das inspeções.

A necessidade de modernizar tais inspeções torna-se evidente ao analisar seus dois principais impactos: na segurança operacional e na eficiência econômica. Primeiramente, a integridade dos processos de MRO está diretamente ligada à prevenção de incidentes. Erros de manutenção são um fator contribuinte significativo em acidentes aéreos, tornando a precisão e a consistência das inspeções um requisito inegociável (SKYbrary, 2022). Por outro lado, o impacto econômico é substancial. O mercado global de MRO está projetado para atingir mais de 100 bilhões de dólares nos próximos anos, e o tempo em que uma aeronave permanece em solo (Aircraft on Ground - AOG) gera custos elevados para as companhias aéreas (Oliver Wyman, 2024). Soluções que aceleram o processo de inspeção de forma confiável podem, portanto, otimizar a disponibilidade da frota e gerar economias significativas.

A aplicação de visão computacional para mitigar os desafios do MRO tem sido um campo de pesquisa ativo, com abordagens focadas em diferentes aspectos da inspeção. Por exemplo, Wiangkam e Jiriwibhakorn (WIANGKAM; JIRIWIBHAKORN, 2024) abordaram a eficiência do gerenciamento do tráfego aéreo ao investigar a detecção de aeronaves em pátios, demonstrando que modelos baseados em visão computacional podem alcançar alta precisão e velocidade. Outra vertente foca na detecção de defeitos na superfície da aeronave. Nesta linha, Suvittawat et al. (SUVITTAWAT et al., 2025) investigaram problemas como arranhões e ferrugem, evidenciando a praticidade do uso de drones de baixo custo para a coleta de dados, embora apontem que os níveis de acurácia ainda necessitam de aprimoramento. Em uma abordagem similar, Merola et al. (MEROLA; GUIDA; MARULO, 2024) investigaram a detecção de danos em componentes específicos, como rebites ausentes ou corroídos, alcançando acurácias relevantes e reforçando o potencial da tecnologia para diminuir o erro humano. Coletivamente, esses estudos fornecem evidências da eficácia dos métodos de visão computacional no ambiente da aviação, ao mesmo tempo

em que destacam a necessidade de datasets mais abrangentes para componentes e danos específicos, abrindo espaço para o desenvolvimento de outros trabalhos.

O objetivo geral deste trabalho é avaliar comparativamente o desempenho de dois algoritmos de detecção de objetos representativos de paradigmas arquiteturais distintos — o **YOLO11s**, baseado em Redes Neurais Convolucionais (CNN), e o **RT-DETR-L**, baseado em *Vision Transformers* —, com foco em identificar e localizar com precisão o trem de pouso de aeronaves em imagens digitais. O estudo visa, ao final, elucidar os *trade-offs* entre os paradigmas e recomendar a abordagem mais adequada conforme os requisitos de cada cenário de aplicação em inspeção aeronáutica. Para alcançar este propósito, os seguintes objetivos específicos foram traçados:

- Estruturar um conjunto de dados de imagens para o problema, com base em um dataset publicamente disponível com anotações supervisionadas pré-existentes, aplicando técnicas de aumento de dados (*data augmentation*) para ampliar a variabilidade e robustez do conjunto.
- Implementar e treinar dois modelos representativos de paradigmas arquiteturais distintos — o **YOLO11s**, baseado em Redes Neurais Convolucionais (CNN), e o **RT-DETR-L**, baseado em *Vision Transformers* — sob condições experimentais idênticas, utilizando o dataset preparado.
- Analisar e comparar o desempenho dos modelos treinados, utilizando métricas quantitativas (como mAP, Precisão, Recall e F1-Score) e qualitativas.

Para alcançar os objetivos propostos, o desenvolvimento deste trabalho foi apoiado por um conjunto de ferramentas e tecnologias consolidadas no campo da visão computacional. O ambiente de desenvolvimento foi baseado na linguagem de programação Python, devido ao seu robusto ecossistema de bibliotecas para ciência de dados e inteligência artificial. Para a preparação dos dados, utilizou-se a plataforma Roboflow, que disponibilizou o dataset público *Landing-Gear IRP* com anotações supervisionadas pré-existentes, ao qual foram aplicadas técnicas de aumento de dados (*data augmentation*), como rotação, ajustes de brilho, contraste e ruído, para enriquecer o conjunto de treinamento. A implementação dos dois modelos se deu por meio do *framework* Ultralytics, com suporte ao *deep learning* via PyTorch. O treinamento foi realizado em ambiente de nuvem, na plataforma Google Colab, com acesso a GPUs NVIDIA Tesla T4 de alto desempenho, essenciais para a execução dos experimentos em tempo hábil.

2 Fundamentação Teórica

Este capítulo apresenta a fundamentação teórica que serve como alicerce para este trabalho. A revisão abrange desde os conceitos gerais de Visão Computacional e Redes Neurais Convolucionais até a tarefa específica de Detecção de Objetos, culminando na análise das principais arquiteturas utilizadas para este fim.

2.1 Aprendizado de Máquina (Machine Learning)

2.1.1 Conceitos Fundamentais

O Aprendizado de Máquina (ML, do inglês Machine Learning) é um subcampo da Inteligência Artificial que se concentra no desenvolvimento de algoritmos capazes de aprender padrões a partir de dados, sem serem explicitamente programados para cada regra de decisão específica (MITCHELL, 1997). Em vez de seguir instruções estáticas, sistemas baseados em ML constroem modelos matemáticos baseados em dados de amostra, conhecidos como "dados de treinamento", para realizar predições ou tomadas de decisão (BISHOP, 2006). Essa capacidade de generalização é fundamental para aplicações onde a complexidade das variáveis torna a programação manual inviável, como é o caso da interpretação visual de cenas complexas em ambientes aeronáuticos.

2.1.2 Aprendizado Supervisionado

Dentro do espectro do ML, este trabalho fundamenta-se no paradigma do Aprendizado Supervisionado. Nesta abordagem, o algoritmo é treinado utilizando um conjunto de dados previamente rotulado, onde para cada entrada x (neste caso, uma imagem do trem de pouso), existe uma saída desejada y correspondente (a localização e classe do componente) (GOODFELLOW; BENGIO; COURVILLE, 2016). O objetivo do modelo é aprender uma função de mapeamento $f : x \rightarrow y$ que minimize o erro entre a predição e o rótulo real. Uma vez treinado, o modelo deve ser capaz de inferir corretamente os rótulos para novos dados não vistos (conjunto de teste), uma propriedade conhecida como capacidade de generalização. Este paradigma é o padrão dominante para tarefas de detecção de objetos modernas, exigindo a curadoria rigorosa de datasets anotados.

2.2 Redes Neurais Artificiais e Deep Learning

2.2.1 O Neurônio Artificial

As Redes Neurais Artificiais (RNAs) são modelos computacionais inspirados na estrutura biológica do cérebro humano. A unidade fundamental de processamento é o neurônio artificial, ou perceptron, que recebe múltiplos sinais de entrada, pondera-os através de pesos sinápticos (w), adiciona um viés (b) e submete o resultado a uma função de ativação não linear (HAYKIN, 2009).

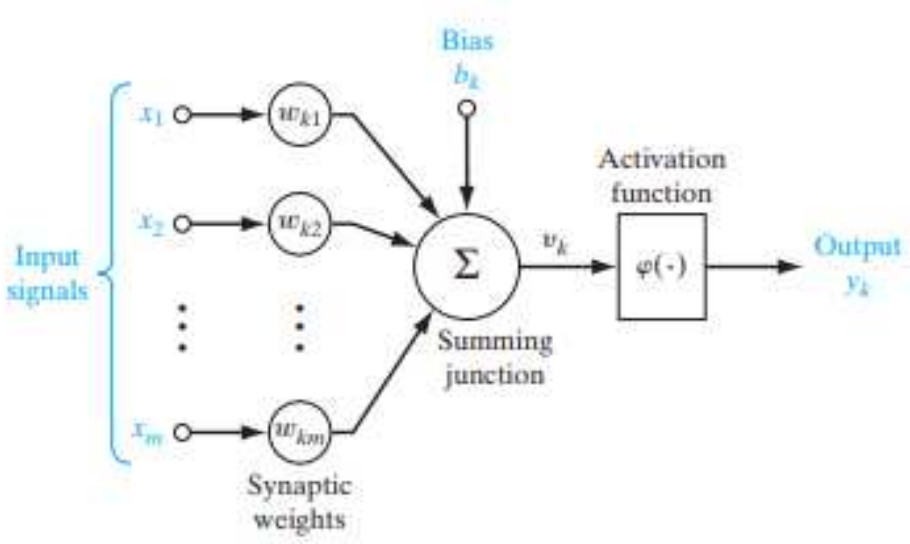


Figura 1 – Modelo não linear de um neurônio artificial. Adaptado de Haykin (2009).

Matematicamente, a saída y de um neurônio pode ser expressa pela Equação 2.1:

$$y = \phi\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{2.1}$$

Na Equação 2.1, ϕ representa a função de ativação (como ReLU ou Sigmoid), essencial para introduzir não-linearidade ao sistema, permitindo que a rede aprenda fronteiras de decisão complexas.

2.2.2 Deep Learning e Redes Multicamadas

Embora o neurônio artificial individual seja capaz de resolver problemas linearmente separáveis, sua capacidade de modelagem é limitada quando isolado (HAYKIN, 2009). O verdadeiro potencial das redes neurais emerge quando múltiplos neurônios são organizados em estruturas hierárquicas compostas por camadas empilhadas, dando origem ao que se conhece como Deep Learning ou Aprendizado Profundo (LECUN; BENGIO; HINTON, 2015).

Diferente das redes superficiais (shallow networks), que tipicamente possuem apenas uma camada oculta, as redes profundas caracterizam-se pela presença de múltiplas camadas ocultas interpostas entre a camada de entrada e a camada de saída. Essa profundidade arquitetural é crucial, pois permite que a rede aprenda representações de dados em múltiplos níveis de abstração (GOODFELLOW; BENGIO; COURVILLE, 2016). Em uma abordagem hierárquica, as primeiras camadas da rede são responsáveis por detectar padrões simples e locais nos dados brutos, enquanto as camadas subsequentes combinam essas características para formar representações progressivamente mais complexas e semânticas.

O treinamento eficaz dessas arquiteturas profundas tornou-se viável graças à aplicação do algoritmo de Backpropagation (retropropagação). Este algoritmo utiliza a regra da cadeia do cálculo diferencial para calcular o gradiente da função de perda em relação a cada peso da rede, propagando o erro da saída de volta para a entrada. Dessa forma, os pesos sinápticos são ajustados iterativamente para minimizar a discrepância entre a predição da rede e o valor real, permitindo que o modelo aprenda a mapear entradas complexas para saídas desejadas com alta precisão (RUMELHART; HINTON; WILLIAMS, 1986).

2.3 Redes Neurais Convolucionais (CNNs)

As Redes Neurais Convolucionais (CNNs) são uma classe de modelos de *deep learning* que se tornaram a principal ferramenta para a maioria das tarefas de visão computacional, demonstrando um potencial notável na identificação e classificação de objetos (MEROLA; GUIDA; MARULO, 2024; LI et al., 2022). Inspiradas no córtex visual humano, as CNNs são projetadas para aprender hierarquias de características visuais de forma automática, utilizando operações especializadas, como a convolução, para extrair padrões que vão do simples (bordas e cores) ao complexo (formas e objetos) (LI et al., 2022).

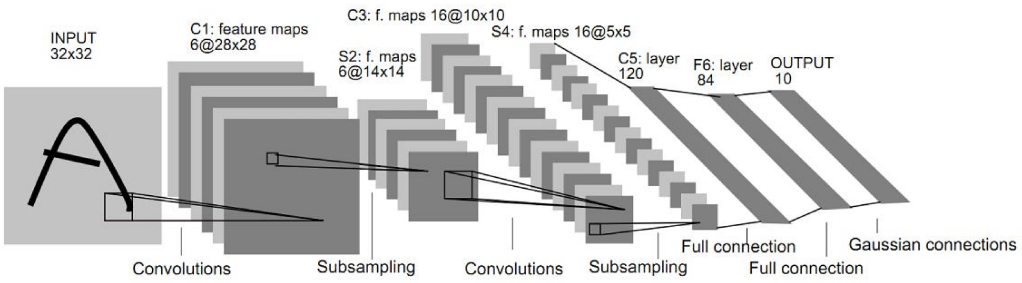


Figura 2 – Arquitetura clássica da LeNet-5, um marco na evolução das CNNs. Adaptado de (LECUN et al., 1998).

2.3.1 Arquitetura e Camadas Principais

O poder de uma CNN reside na sua arquitetura hierárquica, que permite a extração progressiva de padrões espaciais. Como ilustrado pela arquitetura clássica *LeNet-5* (Figura 2), o processamento inicia-se com uma imagem de entrada (neste caso, de 32×32 pixels) e segue um fluxo estruturado em três tipos de camadas fundamentais:

- **Camadas Convolucionais (C):** Constituem o núcleo da rede. Elas aplicam filtros (*kernels*) que realizam operações de convolução sobre a entrada para extrair mapas de características (*feature maps*). Na Figura 2, observa-se que a camada **C1** gera 6 mapas de 28×28, enquanto a **C3** aprofunda a extração para 16 mapas de 10×10, codificando informações cada vez mais abstratas (MEROLA; GUIDA; MARULO, 2024).
- **Camadas de Subamostragem ou Pooling (S):** Intercaladas entre as convoluções, estas camadas (como **S2** e **S4**) reduzem as dimensões espaciais dos mapas de características. Este processo diminui a quantidade de parâmetros computacionais e confere à rede invariância a pequenas translações e distorções na imagem de entrada. Nota-se que a resolução é reduzida pela metade em cada etapa (ex: de 28×28 para 14×14).
- **Camadas Totalmente Conectadas (F):** Localizadas ao final da rede (**C5** e **F6**), estas camadas perdem a estrutura espacial e tratam os dados como vetores unidimensionais. Elas recebem as características de alto nível extraídas pelas camadas anteriores para realizar o raciocínio lógico final.
- **Camada de Saída (*Output*):** A etapa final utiliza funções de ativação (como conexões Gaussianas ou *Softmax*) para gerar as probabilidades de classificação. No exemplo da *LeNet-5*, a saída possui 10 neurônios, correspondendo aos dígitos de 0 a 9.

2.3.2 Aprendizagem por Transferência (Transfer Learning)

A Aprendizagem por Transferência é uma técnica fundamental no *deep learning* moderno, que permite aproveitar o conhecimento adquirido por uma rede neural em uma tarefa para melhorar o desempenho em outra tarefa relacionada (GOODFELLOW; BENGIO; COURVILLE, 2016). No contexto de visão computacional, essa abordagem consiste em utilizar os pesos de uma rede pré-treinada em um grande conjunto de dados genérico (como o ImageNet ou COCO) e adaptá-los para um problema específico através do ajuste fino (*fine-tuning*).

O processo de *Transfer Learning* baseia-se na premissa de que as características de baixo nível aprendidas pela rede (como detectores de bordas, texturas e gradientes) são

transferíveis entre diferentes domínios visuais. Assim, apenas as camadas superiores da rede, responsáveis por características mais específicas da tarefa, precisam ser retreinadas com o novo conjunto de dados (MEROLA; GUIDA; MARULO, 2024).

As principais vantagens dessa abordagem incluem:

- **Redução da necessidade de dados:** Datasets menores podem ser utilizados efetivamente, pois a rede já possui conhecimento visual prévio;
- **Convergência mais rápida:** O treinamento requer menos épocas, economizando tempo e recursos computacionais;
- **Melhor generalização:** Modelos pré-treinados em datasets diversos tendem a generalizar melhor para novos cenários;
- **Prevenção de overfitting:** Especialmente importante quando o dataset de treinamento é limitado.

Neste trabalho, a estratégia de *Transfer Learning* foi aplicada utilizando pesos pré-treinados na base de dados COCO, que contém mais de 200.000 imagens com 80 classes de objetos diversos, fornecendo uma base sólida de conhecimento visual para a detecção de trens de pouso.

2.4 Visão Computacional

Após compreender os fundamentos das Redes Neurais Convolucionais e as técnicas para seu treinamento eficiente, é essencial contextualizar sua aplicação no campo mais amplo da Visão Computacional, área que fornece o arcabouço teórico e prático para este trabalho.

A Visão Computacional é um campo interdisciplinar da ciência da computação e da inteligência artificial que busca desenvolver técnicas para que os computadores possam "ver", processar e extrair informações significativas de imagens e vídeos digitais, buscando representar de forma semelhante a visão humana (SZELISKI, 2022). Em sua essência, a visão computacional não se limita ao processamento de imagens — cujo objetivo é transformar uma imagem em outra, por exemplo, ajustando o brilho — mas foca na interpretação do conteúdo visual para tomar decisões ou realizar tarefas. As principais tarefas da área incluem a classificação, que identifica o que está em uma imagem; a detecção de objetos, que localiza e classifica múltiplos objetos; e a segmentação, que delimita a fronteira exata de cada objeto em nível de pixel (VOULODIMOS et al., 2018).

O campo da visão computacional possui um histórico de evolução marcante. Durante os anos de 1990 e 2000, a área foi dominada por métodos que dependiam da extração

manual de características visuais, como SIFT (Scale-Invariant Feature Transform) e HOG (Histogram of Oriented Gradients) (VOULODIMOS et al., 2018). Em meados de 2012, o campo foi transformado com a introdução de redes neurais profundas, o que deu início à era do *deep learning* e revolucionou a forma como os problemas de visão são resolvidos, superando massivamente as técnicas anteriores (KRIZHEVSKY; SUTSKEVER; HINTON, 2017).

Recentemente, os avanços na área têm sido exponenciais. A evolução de arquiteturas de redes neurais cada vez mais eficientes permitiu a execução de tarefas complexas em tempo real, viabilizando aplicações em dispositivos com poder computacional limitado, como drones e smartphones (SUVITTAWAT et al., 2025). As aplicações da visão computacional são vastas, abrangendo desde o setor automotivo, com os veículos autônomos, até a área da saúde, com o auxílio ao diagnóstico médico (VOULODIMOS et al., 2018). Nesse contexto, a aplicação na Manutenção, Reparo e Operações (MRO) aeronáutico, foco deste trabalho, surge como uma extensão natural, utilizando a visão computacional para aprimorar a segurança e a eficiência das inspeções de componentes críticos (MEROLA; GUIDA; MARULO, 2024; SUVITTAWAT et al., 2025).

2.5 Detecção de Objetos

Dentre as diversas tarefas da visão computacional, a Detecção de Objetos é a que melhor se adequa ao problema proposto neste trabalho, pois permite não apenas identificar a presença do trem de pouso, mas também localizar precisamente sua posição na imagem.

A Detecção de Objetos é uma tarefa fundamental da visão computacional, que visa responder a duas perguntas sobre uma imagem: "Quais objetos estão presentes?"e "Onde eles estão localizados?"(ZOU et al., 2023). Diferentemente da classificação, a detecção identifica múltiplas instâncias de objetos, envolvendo cada uma em uma caixa delimitadora (*bounding box*) e atribuindo-lhe um rótulo de classe.

O sucesso dessa tarefa é medido por métricas rigorosas. A Interseção sobre União (IoU) avalia a precisão da localização da caixa delimitadora. Com base nela, são calculadas a Precisão (*Precision*) e a Revocação (*Recall*). A métrica consolidada para avaliar o desempenho geral do modelo é a Média de Precisão Média (mAP), considerada o padrão-ouro para comparar diferentes detectores (SUVITTAWAT et al., 2025; ZOU et al., 2023).

Historicamente, as abordagens para detecção evoluíram de métodos clássicos para duas grandes famílias no *deep learning*: os modelos de duas etapas (como a família R-CNN), que primeiro propõem regiões e depois as classificam; e os modelos de estágio único (como o YOLO), que realizam a localização e classificação em uma única passagem, permitindo a detecção em tempo real (SUVITTAWAT et al., 2025; ZOU et al., 2023).

Os avanços mais recentes, como o YOLO11 e o RT-DETR, continuam a aprimorar essas técnicas, tornando-as cada vez mais precisas e eficientes para aplicações industriais, como a detecção de defeitos em aeronaves (SUVITTAWAT et al., 2025; MEROLA; GUIDA; MARULO, 2024).

2.6 Arquiteturas para Detecção de Objetos

Para compreender as abordagens investigadas neste trabalho, é fundamental contextualizar as arquiteturas de detecção de objetos dentro da evolução histórica do campo.

A evolução dos detectores baseados em *deep learning* consolidou três paradigmas principais: os de **duas etapas** (*two-stage*), como a família R-CNN, que propõem regiões candidatas antes de classificá-las, oferecendo alta precisão ao custo de velocidade de inferência; os de **estágio único** (*single-stage*), como a família YOLO, que realizam localização e classificação em uma única passagem pela rede, viabilizando a detecção em tempo real; e os baseados em ***Transformers***, como a família DETR, que substituem as operações locais das CNNs por mecanismos globais de atenção, eliminando componentes heurísticos como *anchor boxes* e NMS. Este trabalho investiga os dois últimos paradigmas, representados pelo **YOLO11s** (estágio único, CNN) e pelo **RT-DETR-L** (*Transformer*), cujos compromissos entre precisão, cobertura e eficiência computacional são diretamente relevantes para aplicações de inspeção aeronáutica. As subseções a seguir detalham a arquitetura de cada um.

2.6.1 Detectores de Estágio Único: A Família YOLO

A família de modelos YOLO (*You Only Look Once*) representa o estado da arte em detectores de estágio único, reconhecida por seu balanço entre eficiência e acurácia (SUVITTAWAT et al., 2025). A filosofia do YOLO é tratar a detecção de objetos como um único problema de regressão, processando a imagem de uma só vez para prever simultaneamente as caixas delimitadoras e as probabilidades de classe (MEROLA; GUIDA; MARULO, 2024). Sua arquitetura é tipicamente dividida em Backbone (uma CNN para extração de características), Neck (que combina características de diferentes escalas) e Head (que realiza as predições finais), uma estrutura que permite alta velocidade de inferência, ideal para aplicações em tempo real (MEROLA; GUIDA; MARULO, 2024).

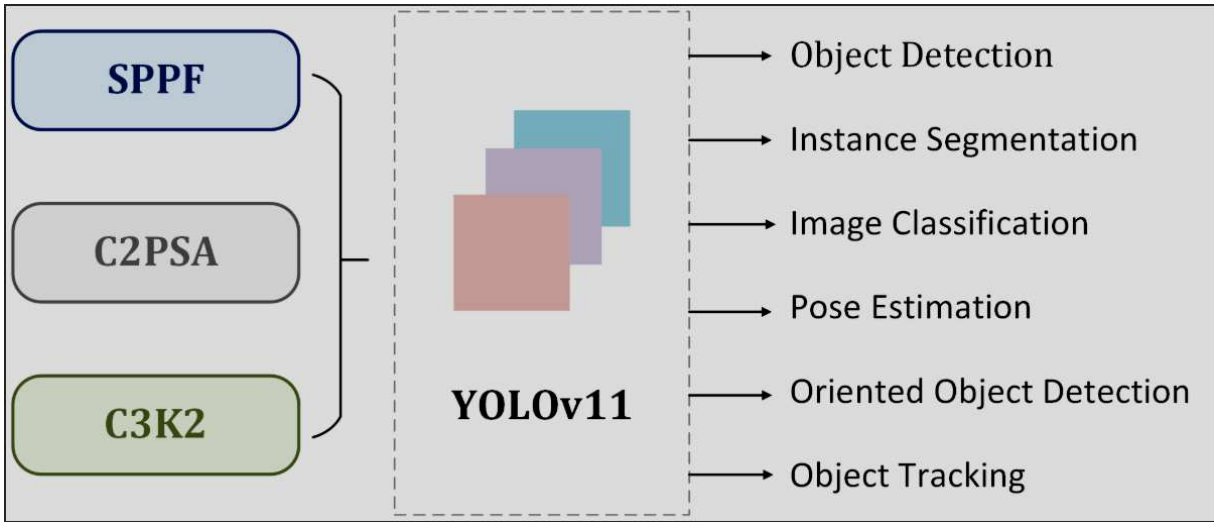


Figura 3 – Visão geral da arquitetura YOLO11, mostrando os componentes principais: Backbone, Neck e Head. Fonte: (KHANAM; HUSSAIN, 2024).

2.6.1.1 Backbone: Extração Hierárquica de Características

O Backbone constitui a espinha dorsal da rede, responsável pela extração progressiva de representações visuais em múltiplos níveis de abstração. A entrada do sistema é uma imagem padronizada em resolução de 640 × 640 pixels, que atravessa uma sequência de camadas convolucionais e blocos especializados.

O fluxo de processamento inicia-se com duas camadas convolucionais sequenciais que realizam a redução gradual da resolução espacial (de 640×640 para 320×320 e posteriormente para 160×160 pixels), enquanto aumentam a profundidade dos mapas de características. Essa estratégia de *downsampling* progressivo permite que a rede capture padrões em escalas crescentes, desde bordas e texturas simples até formas geométricas complexas.

Um elemento arquitetural distintivo do YOLO11 é o bloco C3K2 (Cross Stage Partial with 2 Kernels), uma evolução do módulo C3 presente em versões anteriores da família YOLO. Este componente implementa conexões residuais (shortcuts) que facilitam o fluxo de gradientes durante o treinamento, mitigando o problema de desaparecimento do gradiente em redes profundas (GOODFELLOW; BENGIO; COURVILLE, 2016). Os blocos C3K2 aparecem em múltiplas posições da rede com configurações variadas do parâmetro shortcut, que controla a presença ou ausência de conexões de atalho, permitindo flexibilidade na profundidade efetiva da rede.

A progressão do Backbone segue uma hierarquia de cinco níveis de resolução (160×160, 80×80, 40×40, 20×20), cada um capturando informações em diferentes escalas espaciais. Esta arquitetura piramidal é fundamental para a capacidade do modelo em detectar objetos de tamanhos variados, desde componentes pequenos até estruturas maiores do trem de pouso.

2.6.1.2 Neck: Fusão Multi-Escala de Características

O módulo Neck desempenha a função crítica de integrar as características extraídas em diferentes níveis de resolução pelo Backbone, implementando uma arquitetura de fusão bidirecional inspirada nos conceitos de Feature Pyramid Network (FPN) e Path Aggregation Network (PAN).

O processamento no Neck inicia-se com o módulo SPPF (Spatial Pyramid Pooling - Fast), posicionado na camada mais profunda (20×20). O SPPF aplica operações de pooling em múltiplas escalas espaciais, capturando informações contextuais em diferentes campos receptivos sem aumentar significativamente o custo computacional. Esta técnica permite que a rede compreenda o contexto global da imagem, essencial para distinguir componentes aeronáuticos em cenários complexos.

Após o SPPF, as características da escala mais profunda (20×20) são enriquecidas pelo módulo **C2PSA** (*C2 with Position-Sensitive Attention*), que aplica um mecanismo de atenção sobre estas representações densas. O C2PSA permite que o modelo foque seletivamente nas regiões de maior relevância semântica antes que estas características alimentem as etapas subsequentes de fusão, contribuindo para a localização precisa de componentes como o trem de pouso.

Subsequentemente, o Neck implementa um fluxo de processamento bidirecional com as características enriquecidas pelo C2PSA:

Caminho Ascendente (*Top-Down Pathway*): Através de operações de *Upsample*, características semanticamente ricas da escala mais profunda são progressivamente propagadas para os níveis de maior resolução. Em cada etapa, estas características são concatenadas (*Concat*) com os mapas correspondentes do Backbone através de conexões laterais, combinando informações semânticas de alto nível com detalhes espaciais de baixo nível.

Caminho Descendente (*Bottom-Up Pathway*): Operações de convolução com *stride* 2 realizam o re-escalonamento das características de alta resolução para escalas menores. As concatenações com mapas das escalas correspondentes garantem a preservação das informações semânticas em toda a pirâmide de características.

As operações de concatenação estrategicamente posicionadas em ambos os caminhos garantem que cada nível de características contenha tanto informações locais detalhadas quanto contexto semântico global, uma propriedade essencial para detecção robusta em múltiplas escalas.

2.6.1.3 Head: Predição Multi-Escala

O módulo Head constitui a etapa final da arquitetura, responsável pela geração das predições de detecção. Diferentemente de abordagens que utilizam uma única camada

de saída, o YOLO11 implementa três Detection Heads paralelos, cada um operando em uma resolução espacial distinta:

- Detect 1 (80×80 pixels): Especializado na detecção de objetos pequenos. A alta resolução espacial preserva detalhes finos, permitindo a identificação de componentes menores ou distantes do trem de pouso.
- Detect 2 (40×40 pixels): Otimizado para objetos de tamanho médio, representando o equilíbrio entre resolução espacial e campo receptivo semântico.
- Detect 3 (20×20 pixels): Focado em objetos grandes. A menor resolução espacial é compensada por um campo receptivo amplo, adequado para a detecção de estruturas maiores.

Cada Head de detecção realiza simultaneamente duas tarefas de regressão e classificação:

- Localização: Predição das coordenadas (x, y, w, h) da caixa delimitadora (*bounding box*).
- Classificação: Distribuição de probabilidades sobre as classes de interesse (no caso deste trabalho, a classe *landing_gear*). Diferentemente das versões anteriores da família YOLO, o YOLO11 não possui um *objectness score* independente: o score de confiança da detecção é derivado diretamente das probabilidades de classe, simplificação introduzida a partir do YOLOv8.

A arquitetura multi-escala permite que o modelo seja invariante ao tamanho do objeto, uma propriedade essencial para aplicações aeronáuticas, onde o trem de pouso pode aparecer em diferentes escalas dependendo da distância de captura, do ângulo da aeronave e da perspectiva da câmera.

2.6.1.4 Eficiência Computacional e Escalabilidade

Uma característica distintiva da família YOLO11 é a disponibilização de múltiplas variantes escaláveis (Nano, Small, Medium, Large, Extra-Large), cada uma representando um ponto diferente no espectro precisão-eficiência. A variante YOLO11s (Small), selecionada para este trabalho, foi projetada para oferecer um equilíbrio otimizado entre acurácia de detecção e velocidade de inferência, tornando-a particularmente adequada para aplicações que exigem processamento em tempo real com recursos computacionais moderados, como sistemas embarcados em UAVs (JOCHER; QIU, 2024).

A modularidade da arquitetura permite que ajustes na profundidade dos blocos C3K2 e na largura dos canais de características (width multiplier) resultem em diferentes

configurações de modelo, viabilizando o deployment em uma ampla gama de dispositivos, desde servidores em nuvem com GPUs de alto desempenho até dispositivos de borda (edge devices) com capacidade computacional limitada.

2.6.2 Vision Transformers (ViT) e Mecanismos de Atenção

2.6.2.1 Do Paradigma Convolucional aos Transformers

Historicamente, a visão computacional foi dominada por arquiteturas baseadas em Redes Neurais Convolucionais (CNNs), que processam imagens através de operações localizadas em janelas espaciais limitadas. Embora eficientes, as CNNs possuem uma limitação inerente: seu campo receptivo é construído progressivamente através de camadas empilhadas, dificultando a captura de dependências de longo alcance sem aumentar substancialmente a profundidade da rede (DOSOVITSKIY et al., 2021).

Em contrapartida, a arquitetura *Transformer*, originalmente proposta por (VASWANI et al., 2023) para processamento de linguagem natural, fundamenta-se no mecanismo de *Self-Attention* (Auto-atenção), que permite relacionar diferentes posições de uma sequência independentemente da distância entre elas. (DOSOVITSKIY et al., 2021) demonstraram que esta capacidade de modelagem global pode ser adaptada para a visão computacional através dos *Vision Transformers* (ViT), tratando uma imagem como uma sequência de *patches* (recortes) processados por camadas de atenção.

A família DETR (*DEtection TRansformer*), introduzida por (CARION et al., 2020), representa a aplicação pioneira de *Transformers* para detecção de objetos, eliminando componentes artesanais como *Non-Maximum Suppression* (NMS) e *anchor boxes* através de uma abordagem *end-to-end*. O RT-DETR (*Real-Time Detection Transformer*) constitui uma evolução desta arquitetura, projetada especificamente para atender aos requisitos de velocidade de inferência necessários para aplicações práticas (ZHAO et al., 2023).

2.6.2.2 Arquitetura do RT-DETR: Visão Geral

Conforme ilustrado na Figura 4, a arquitetura do RT-DETR é estruturada em quatro componentes funcionais principais: o *Backbone* para extração de características, o *Efficient Hybrid Encoder* para enriquecimento multi-escala, o módulo *IoU-aware Query Selection* para inicialização inteligente de *queries*, e o *Decoder & Head* para geração das predições finais.

Diferentemente do YOLO, que prediz detecções diretamente sobre uma grade espacial densa, o RT-DETR adota o paradigma de detecção baseada em *queries*: um conjunto fixo de vetores aprendíveis (*object queries*) interroga as características da imagem através de mecanismos de atenção, e cada *query* é responsável por predizer um objeto potencial.

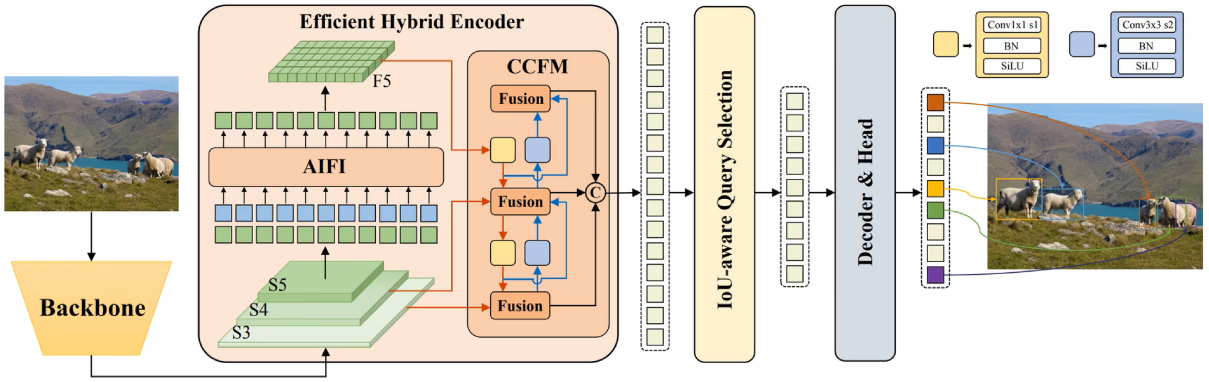


Figura 4 – Visão geral da arquitetura RT-DETR, mostrando os componentes principais: Backbone, Efficient Hybrid Encoder (AIFI e CCFM), IoU-aware Query Selection e Decoder & Head. Fonte: (ZHAO et al., 2023).

Esta abordagem elimina a necessidade de heurísticas como NMS, tornando o *pipeline* verdadeiramente diferenciável de ponta a ponta.

2.6.2.3 Backbone: Extração de Características Multi-Escala

Similar ao YOLO11, o RT-DETR utiliza uma CNN como *Backbone* para extrair representações hierárquicas da imagem de entrada. A Figura 4 ilustra a geração de mapas de características em múltiplas escalas (S3, S4, S5), cada uma capturando informações em diferentes níveis de abstração e resolução espacial.

A escolha de uma CNN como *Backbone*, em vez de um ViT puro, justifica-se por considerações de eficiência: convoluções são altamente otimizadas em *hardware* moderno (GPUs) e fornecem um bom balanceamento entre custo computacional e qualidade das características iniciais para o processamento subsequente pelos *Transformers*.

2.6.2.4 Efficient Hybrid Encoder: Fusão Inteligente de Características

O *Efficient Hybrid Encoder* representa a inovação central do RT-DETR, combinando duas estratégias complementares de processamento de características:

2.6.2.4.1 AIFI: Attention-based Intrascale Feature Interaction

O módulo AIFI implementa *Self-Attention* dentro de cada escala de características (intrascala). Para a escala mais profunda (F5, correspondente a S5 do *Backbone*), o AIFI permite que cada posição espacial "observe" todas as outras posições daquela mesma escala, capturando relações contextuais globais.

Matematicamente, o mecanismo de *Self-Attention* é definido pela Equação 2.2:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.2}$$

Na Equação 2.2, Q (*queries*), K (*keys*) e V (*values*) são projeções lineares das características de entrada, e d_k é a dimensionalidade das *keys*. Esta operação permite que o modelo identifique regiões semanticamente relacionadas, mesmo que espacialmente distantes, uma capacidade crucial para contextos aeronáuticos onde componentes do trem de pouso podem estar parcialmente oclusos ou em ângulos incomuns.

2.6.2.4.2 CCFM: Cross-scale Feature Mixing with Fusion

O componente CCFM (*Cross-scale Feature Mixing*) realiza a fusão de características entre diferentes escalas (*cross-scale*). Observando a Figura 4, nota-se que o CCFM contém três blocos de *Fusion* que integram características de escalas adjacentes através de operações de concatenação (C) e transformações convolucionais. O detalhe interno destes blocos de *Fusion*, visível no diagrama, revela camadas de Conv1x1 (projeções lineares), *Batch Normalization* (BN) e SiLU (função de ativação *Sigmoid Linear Unit*), responsáveis pelo refinamento local das características fundidas.

Esta estratégia híbrida é análoga ao *Neck* do YOLO11, mas com uma diferença fundamental: enquanto o YOLO utiliza concatenações e convoluções padrão, o RT-DETR intercala operações de atenção (AIFI) com a fusão espacial (CCFM), permitindo tanto interação global quanto refinamento local.

A saída do *Efficient Hybrid Encoder* é um conjunto enriquecido de características multi-escala que combina:

- Informação semântica global (via AIFI)
- Detalhes espaciais locais (via convoluções no CCFM)
- Integração multi-escala (via *Fusion blocks*)

2.6.2.5 IoU-aware Query Selection: Inicialização Inteligente

Um desafio dos *Transformers* de detecção é a inicialização aleatória das *object queries*, que pode resultar em convergência lenta. O RT-DETR introduz o módulo *IoU-aware Query Selection* para endereçar esta limitação.

Este componente analisa as características codificadas e seleciona um subconjunto de posições espaciais que possuem alta probabilidade de conter objetos (baseado em *scores* de IoU preditos). Estas posições são então utilizadas para inicializar as *queries* do *decoder* de forma mais informada, acelerando significativamente o treinamento e melhorando a *performance* final.

O mecanismo funciona como um "filtro de atenção" que direciona o *decoder* para focar em regiões promissoras, reduzindo o espaço de busca e tornando o processo mais eficiente.

2.6.2.6 Decoder & Head: Predição via Atenção Cruzada

O módulo *Decoder* do RT-DETR é composto por múltiplas camadas de *Transformer Decoder* empilhadas. Cada camada executa duas operações principais:

- **Self-Attention nas Queries**: As *object queries* interagem entre si para garantir que cada uma seja responsável por um objeto distinto, evitando detecções duplicadas.
- **Cross-Attention (Atenção Cruzada)**: Cada *query* "consulta" as características da imagem codificadas, extraindo informações relevantes para sua predição específica. Esta é a operação que efetivamente "localiza" os objetos na imagem.

Finalmente, o *Head* de detecção processa cada *query* refinada e produz três predições:

- Coordenadas da caixa delimitadora (x, y, w, h)
- *Score* de confiança (probabilidade de existência do objeto)
- Distribuição de probabilidades de classe

Como pode ser observado na Figura 4, diferentemente do YOLO, que possui múltiplos *heads* em diferentes escalas espaciais, o RT-DETR utiliza um único *head* que opera sobre *queries* aprendíveis, tornando a arquitetura mais unificada e interpretável.

2.6.2.7 Vantagens e Compromissos do Paradigma Transformer

A arquitetura RT-DETR apresenta vantagens distintas em relação aos detectores convolucionais:

Modelagem Global: O mecanismo de atenção permite capturar dependências de longo alcance desde as primeiras camadas, tornando o modelo mais robusto a oclusões parciais e cenários com contexto complexo.

Arquitetura End-to-End: A eliminação de componentes heurísticos como NMS e *anchor boxes* torna o *pipeline* totalmente diferenciável, facilitando o treinamento e a otimização.

Interpretabilidade: Os mapas de atenção podem ser visualizados para entender quais regiões da imagem influenciaram cada detecção, oferecendo maior transparência.

Contudo, estas vantagens vêm acompanhadas de compromissos:

Custo Computacional: A operação de *Self-Attention* possui complexidade quadrática $O(n^2)$ em relação ao número de posições espaciais, tornando-a mais custosa que convoluções locais.

Requisitos de Memória: O armazenamento das matrizes de atenção exige maior quantidade de memória, o que pode limitar o tamanho do lote (*batch*) durante o treinamento.

Otimização para Tempo Real: Embora denominado "*Real-Time*", o RT-DETR geralmente apresenta velocidade de inferência inferior aos modelos YOLO de tamanho comparável, *trade-off* discutido no Capítulo 5.

2.7 Métricas de Avaliação de Desempenho

A validação quantitativa de um modelo de detecção de objetos é substancialmente mais complexa do que em tarefas de classificação simples. Não basta saber se a classe predita está correta; é necessário verificar se a localização do objeto na imagem é precisa. Para avaliar a eficácia das arquiteturas propostas neste trabalho, utilizam-se métricas padronizadas pela comunidade científica e competições de referência, como o COCO Benchmark.

2.7.1 Intersection over Union (IoU)

A base para todas as métricas de detecção é o Intersection over Union (IoU), um coeficiente geométrico que quantifica a sobreposição entre a caixa delimitadora predita pelo modelo (B_p) e a caixa real anotada no dataset (B_{gt} - ground truth). O IoU é definido pela razão entre a área da interseção e a área da união das duas caixas, conforme a Equação 2.3 (PADILLA; NETTO; SILVA, 2020):

$$IoU = \frac{\text{Área}(B_p \cap B_{gt})}{\text{Área}(B_p \cup B_{gt})} \tag{2.3}$$

Este valor varia de 0 (sem sobreposição) a 1 (sobreposição perfeita). Para fins de avaliação, define-se um limiar de aceitação (geralmente $\alpha = 0.5$). Se $IoU \geq \alpha$, a detecção é considerada correta (Verdadeiro Positivo); caso contrário, é considerada um erro de localização.

2.7.2 Precisão, Revocação e F1-Score

Com base no critério do IoU, as predições são classificadas em três categorias fundamentais:

Verdadeiros Positivos (TP): Detecção correta com $IoU \geq \alpha$.

Falsos Positivos (FP): Detecção de um objeto onde não existe (ou $IoU < \alpha$). No contexto de MRO, isso representaria o sistema identificar um trem de pouso em uma área vazia.

Falsos Negativos (FN): Falha em detectar um objeto existente. Este é o erro mais crítico em inspeções de segurança, pois implica deixar passar um componente que deveria ser analisado.

A partir dessas definições, derivam-se as métricas de primeira ordem:

Precisão (Precision): A proporção de detecções positivas que são realmente corretas, dada pela Equação 2.4.

$$P = \frac{TP}{TP + FP} \tag{2.4}$$

Revocação (Recall): A proporção de objetos reais que o modelo foi capaz de encontrar, dada pela Equação 2.5.

$$R = \frac{TP}{TP + FN} \tag{2.5}$$

Existe um trade-off natural entre essas métricas: aumentar a sensibilidade do modelo para encontrar todos os trens de pouso (alto Recall) tende a aumentar o número de alarmes falsos (baixa Precisão). Para obter uma avaliação unificada, utiliza-se o F1-Score, que é a média harmônica entre Precisão e Revocação, calculada conforme a Equação 2.6:

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{2.6}$$

O F1-Score é particularmente útil em datasets desbalanceados e para aplicações industriais onde se busca um equilíbrio entre confiabilidade e abrangência da inspeção.

2.7.3 Mean Average Precision (mAP)

A métrica global padrão-ouro para comparação de detectores é a Mean Average Precision (mAP). Ela é calculada analisando a curva Precisão-Revocação (PR Curve).

Average Precision (AP): Para uma classe específica, a AP é calculada como a área sob a curva PR interpolada. Isso resume o desempenho do modelo em diferentes limiares de confiança.

Mean Average Precision (mAP): É a média aritmética das APs de todas as classes presentes no dataset, conforme a Equação 2.7.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{2.7}$$

Para uma avaliação robusta, este trabalho reportará o mAP em dois padrões principais, conforme definido pelo benchmark COCO (LIN et al., 2015):

mAP@50: Considera apenas predições com $IoU > 0.5$. É uma métrica mais permissiva, focada na detecção da presença do objeto.

mAP@50-95: Calcula a média do mAP variando o limiar de IoU de 0,50 até 0,95, com passos de 0,05. Esta é uma métrica muito mais rigorosa, que penaliza modelos que encontram o objeto, mas não conseguem delimitar sua caixa com exatidão milimétrica. Para aplicações de robótica autônoma, onde o drone precisa se aproximar do objeto, um alto mAP@50-95 é desejável.

3 Trabalhos Relacionados

Os trabalhos a seguir foram selecionados com base em três dimensões de aderência ao presente estudo: (i) aplicação de visão computacional no contexto de manutenção e inspeção aeronáutica (domínio); (ii) uso de arquiteturas de detecção de objetos da família YOLO ou baseadas em *Transformers* (metodologia); e (iii) avaliação de *trade-offs* entre desempenho e eficiência computacional (análise comparativa). Em conjunto, os três trabalhos fornecem um panorama do estado da arte que contextualiza e fundamenta as escolhas metodológicas desta pesquisa.

O trabalho de Suvittawat et al. (SUVITTAWAT et al., 2025) é o mais diretamente alinhado a esta pesquisa. Seu objetivo foi revisar o estado atual dos algoritmos de visão computacional para a detecção de defeitos em aeronaves e, ao mesmo tempo, avaliar experimentalmente a viabilidade de soluções de baixo custo para essa tarefa. A metodologia envolveu a coleta de um novo dataset público de defeitos, como ferrugem e arranhões, utilizando um drone para capturar imagens de um Boeing 747. Em seguida, os autores treinaram e compararam o desempenho dos modelos YOLOv9 e RT-DETR — os mesmos paradigmas arquiteturais (CNN vs. *Transformer*) investigados neste TCC, embora com versões distintas. Os resultados indicaram desempenho geral comparável entre os modelos, com scores de mAP@50 entre 0,70 e 0,75, com o YOLOv9 apresentando ligeira vantagem em acurácia e velocidade de inferência e o RT-DETR demonstrando convergência de treinamento mais rápida. Esses valores servem como referência direta para a avaliação dos resultados obtidos neste trabalho.

Em uma abordagem mais focada na detecção de componentes específicos, o trabalho de Merola et al. (MEROLA; GUIDA; MARULO, 2024) desenvolveu um método de visão computacional para a identificação de danos em fuselagens de aeronaves em escala real. Os autores utilizaram a arquitetura YOLOv8 sobre um dataset próprio de 964 imagens com mais de 6.000 anotações de defeitos — rebites ausentes, rebites corroídos e danos gerais — capturadas em laboratório com diferentes condições de iluminação simuladas. Os resultados demonstraram acurácias de 75% para rebites ausentes, 63% para rebites corroídos e 87% para danos gerais. A relevância deste estudo reside na analogia metodológica com o presente TCC: ambos aplicam modelos da família YOLO à detecção de um componente aeronáutico específico em imagens reais, utilizando *data augmentation* para compensar o tamanho reduzido do dataset.

O trabalho de Wiangkam e Jiriwibhakorn (WIANGKAM; JIRIWIBHAKORN, 2024) investigou a detecção automática de aeronaves em pátios de aeroportos, comparando cinco variantes de tamanho do modelo YOLOv8 (Nano, Small, Medium, Large e Extra-

Large) sobre um dataset de 1.000 imagens. Os resultados quantificaram com precisão o *trade-off* entre desempenho e eficiência: a variante maior (YOLOv8x) alcançou precisão de 0,94, enquanto a menor (YOLOv8n) processou cada imagem em apenas 0,95 milissegundos. Embora o objeto de detecção (aeronave inteira) seja mais simples do que o trem de pouso, este estudo é valioso pela metodologia comparativa adotada: a avaliação sistemática de variantes de um mesmo modelo em função do compromisso precisão-velocidade é análoga à análise comparativa entre YOLO11s e RT-DETR-L conduzida neste trabalho.

A análise conjunta destes três trabalhos revela um padrão comum: modelos baseados na família YOLO consistentemente demonstram viabilidade para inspeção aeronáutica automatizada, atingindo acurácias entre 63% e 94% dependendo da complexidade da tarefa e do componente inspecionado. Ao mesmo tempo, a comparação de Suvittawat et al. evidencia que arquiteturas baseadas em *Transformers* (RT-DETR) constituem uma alternativa competitiva, com dinâmica de convergência distinta. A lacuna identificada na literatura é a ausência de estudos que comparem diretamente esses dois paradigmas — CNN e *Transformer* — aplicados especificamente à detecção do trem de pouso, sob condições experimentais rigorosamente equalizadas. Preencher esta lacuna é o objetivo central do presente trabalho, cujo desenvolvimento metodológico é descrito no capítulo seguinte.

4 Metodologia Proposta

Este capítulo descreve a abordagem metodológica adotada para a análise comparativa de arquiteturas de detecção de objetos aplicadas à identificação de trens de pouso aeronáuticos. Foram implementados e avaliados dois modelos representativos de paradigmas distintos: o **YOLO11s** (*You Only Look Once - Small*), um detector de estágio único baseado em Redes Neurais Convolucionais (CNNs), e o **RT-DETR-L** (*Real-Time Detection Transformer - Large*), uma arquitetura baseada em *Vision Transformers*. Ambos os modelos foram treinados sob condições experimentais idênticas, permitindo uma comparação justa dos *trade-offs* entre precisão, cobertura e eficiência computacional de cada paradigma.

4.1 Ambiente de Desenvolvimento e Hardware

O treinamento de arquiteturas de *Deep Learning* exige alta capacidade de processamento paralelo. Para viabilizar os experimentos, utilizou-se a plataforma de computação em nuvem **Google Colab**, que permite o acesso escalonável a recursos de alto desempenho.

A infraestrutura de *hardware* foi composta por uma GPU **NVIDIA Tesla T4**, cuja capacidade de processamento paralelo e disponibilidade de memória de vídeo foram determinantes para viabilizar o treinamento de ambos os modelos, especialmente o RT-DETR-L, que exige maior carga computacional em razão da complexidade quadrática do mecanismo de *Self-Attention*.

4.2 Base de Dados (*Dataset*)

Para o desenvolvimento e validação dos modelos, utilizou-se o conjunto de dados *Landing-Gear IRP* (Cranfield University, 2024), disponibilizado pela *Cranfield University* via plataforma *Roboflow Universe*. As imagens foram capturadas em cenários operacionais variados, tais como procedimentos de pouso, decolagem e permanência no pátio, possuindo anotações pré-processadas no formato padrão *YOLO*, com caixas delimitadoras (*bounding boxes*) que circundam o objeto de interesse. O conjunto foca em uma única classe alvo: *landing_gear* (trem de pouso).

4.2.1 Pré-processamento e Aumento de Dados (*Data Augmentation*)

O conjunto original do *Landing-Gear IRP* é composto por **1.245 imagens** com anotações supervisionadas pré-existentes. Com o intuito de ampliar a robustez das redes e mitigar o risco de *overfitting*, aplicaram-se técnicas de *Data Augmentation* diretamente na plataforma Roboflow. Este processo multiplicou o volume de dados original, simulando variações de captura e condições ambientais adversas. As transformações aplicadas, conforme especificado na configuração do dataset, incluíram:

- **Auto-Orient:** Correção automática da orientação das imagens baseada nos metadados EXIF;
- **Resize:** Redimensionamento por esticamento (*stretch*) para a resolução padrão de 640×640 pixels;
- **Auto-Adjust Contrast:** Equalização adaptativa de contraste para normalizar as condições de iluminação;
- **Flip:** Espelhamento horizontal para aumentar a variabilidade de orientação;
- **Rotação 90°:** Aplicação de rotações em sentido horário e anti-horário;
- **Crop:** Recorte aleatório entre 0% (mínimo) e 20% (máximo) para simular diferentes enquadramentos;
- **Rotation:** Rotação aleatória entre -15° e +15° para simular inclinações da aeronave;
- **Shear:** Distorção de cisalhamento de ±10° nos eixos horizontal e vertical;
- **Grayscale:** Conversão para escala de cinza em 15% das imagens;
- **Hue:** Variação de matiz entre -15° e +15° para simular diferentes condições de luz;
- **Saturation:** Ajuste de saturação entre -25% e +25%;
- **Brightness:** Oscilação de brilho entre -15% e +15% para mimetizar diferentes condições de iluminação solar;
- **Exposure:** Ajuste de exposição entre -10% e +10%;
- **Blur:** Desfoque gaussiano de até 2,5 px para simular imagens fora de foco;
- **Noise:** Inserção de ruído em até 0,1% dos pixels para simular a granulação típica de sensores de câmeras.

Cada exemplo do conjunto de treinamento foi configurado para gerar 3 variações aumentadas (*outputs per training example*), expandindo significativamente o volume de dados disponíveis para o aprendizado. O *dataset* final totalizou **2.979 imagens**, consolidando uma base de conhecimento mais diversificada e robusta para o treinamento dos modelos de aprendizado profundo.

4.2.2 Distribuição do Conjunto de Dados

A partição dos dados seguiu a divisão padrão estabelecida pela plataforma Roboflow para este dataset específico, garantindo a integridade da avaliação:

- **Treino (87%):** Destinado ao ajuste dos pesos sinápticos, totalizando **2.601 imagens**, composto por imagens originais e suas variações aumentadas;
- **Validação (8%):** Utilizado para o ajuste de hiperparâmetros e monitoramento do desempenho durante o treinamento, contendo **252 imagens**;
- **Teste (4%):** Reservado para a avaliação final do modelo, composto por **126 imagens**, garantindo que a métrica final reflita o desempenho em condições reais de operação.

4.3 Implementação e Execução do Treinamento

A etapa de treinamento foi operacionalizada utilizando o *framework* Ultralytics (JOCHER; QIU, 2024), que provê uma interface unificada de alto nível para o treinamento e avaliação de modelos de detecção de objetos, incluindo tanto a família YOLO quanto o RT-DETR. A estratégia adotada baseou-se no paradigma de *Transfer Learning* (Aprendizagem por Transferência), utilizando pesos pré-treinados na base de dados COCO para ambos os modelos, acelerando a convergência e aumentando a robustez, mitigando as limitações impostas pelo tamanho do conjunto de dados.

4.3.1 Seleção dos Modelos

Para os experimentos, foram selecionadas as arquiteturas **YOLO11s** e **RT-DETR-L**. A escolha do YOLO11s justifica-se pelo equilíbrio entre eficiência computacional e capacidade de representação da variante *Small*: mais precisa que a variante *Nano*, sem as exigências de hardware das variantes maiores. O RT-DETR-L foi selecionado como representante do paradigma *Transformer*, por ser a variante *Large* disponibilizada pela Ultralytics com pesos pré-treinados no COCO, garantindo comparabilidade direta com o YOLO11s em termos de qualidade dos pesos iniciais.

4.3.2 Configuração do Treinamento

Para garantir comparabilidade metodológica rigorosa, ambos os modelos foram treinados com o mesmo conjunto reduzido de parâmetros definidos explicitamente, cabendo ao framework Ultralytics a determinação automática dos demais hiperparâmetros com base nos valores padrão otimizados para cada arquitetura.

Os parâmetros definidos explicitamente em ambos os treinamentos foram:

- **Épocas (`epochs=50`):** 50 ciclos completos sobre o conjunto de dados para ambos os modelos, garantindo equidade nas condições de treinamento;
- **Dimensão da Imagem (`imgsz=640`):** Resolução de 640×640 pixels, padrão da família YOLO e compatível com o RT-DETR;
- **Dataset (`data`):** O arquivo `data.yaml` do dataset *Landing-Gear IRP*, contendo os caminhos para os conjuntos de treino, validação e teste.

Os demais hiperparâmetros foram determinados automaticamente pelo framework Ultralytics. Os principais valores assumidos, extraídos dos arquivos `args.yaml` gerados ao final de cada treinamento, estão documentados na Tabela 1.

Tabela 1 – Principais hiperparâmetros de treinamento determinados automaticamente pelo framework Ultralytics.

Hiperparâmetro	**YOLO11s**	**RT-DETR-L**
Tamanho do Lote (`batch`)	16	16
Otimizador (`optimizer`)	auto (SGD)	auto (AdamW)
Paciência (`patience`)	100	100
Taxa de Aprendizado Inicial (`lr0`)	0.01	0.01
Taxa de Aprendizado Final (`lrf`)	0.01	0.01
Momentum (`momentum`)	0.937	0.937
Weight Decay (`weight_decay`)	0.0005	0.0005
Warm-up (`warmup_epochs`)	3	3
IoU (`iou`)	0.7	0.7
Precisão Mista (`amp`)	True	True

4.3.3 Procedimento de Treinamento e Persistência de Dados

O fluxo de execução foi automatizado via *scripts* Python. Durante o treinamento supervisionado, o algoritmo calculou iterativamente o erro de predição das caixas delimitadoras, atualizando os pesos da rede neural via *backpropagation*.

Considerando a volatilidade do sistema de arquivos das máquinas virtuais em nuvem, integrou-se o diretório de saída ao **Google Drive** via montagem de volume. Esta configuração garantiu a persistência de todos os artefatos gerados, como curvas de

precisão-revocação (*PR Curve*), matrizes de confusão e, fundamentalmente, o arquivo de pesos otimizados (`best.pt`).

4.3.4 Refinamento Iterativo

Os resultados obtidos constituem a base empírica desta pesquisa. A metodologia admite etapas futuras de refinamento, nas quais novos experimentos poderão ser conduzidos mediante a alteração de hiperparâmetros específicos — tais como otimizadores, taxas de aprendizado e técnicas de regularização — ou pela avaliação de variantes arquiteturais ainda não exploradas, como versões maiores (RT-DETR-X) ou menores (YOLO11n) dos modelos aqui investigados, aprofundando o mapeamento do espectro precisão-eficiência.

Os resultados obtidos com ambos os modelos são apresentados e discutidos no capítulo seguinte, onde as métricas quantitativas e as análises qualitativas das predições fundamentam as recomendações finais sobre a aplicabilidade de cada paradigma em cenários de inspeção aeronáutica.

5 Resultados e Discussão

Este capítulo apresenta e analisa os resultados experimentais obtidos no treinamento dos dois modelos de detecção de objetos investigados neste trabalho: o **YOLO11s** (baseado em Redes Neurais Convolucionais) e o **RT-DETR-L** (baseado em *Vision Transformers*). A avaliação é conduzida de forma comparativa e simétrica entre os dois paradigmas, abrangendo as curvas de treinamento, as curvas de avaliação, a análise matricial de erros e a inspeção visual qualitativa das predições, culminando em uma análise comparativa que fundamenta as recomendações finais sobre a aplicabilidade de cada paradigma em cenários de inspeção aeronáutica.

5.1 Configuração Experimental

Para garantir comparabilidade metodológica rigorosa, ambos os modelos foram treinados sob **condições experimentais idênticas**:

- **Plataforma:** Google Colab com GPU NVIDIA Tesla T4 (16 GB VRAM)
- **Dataset:** Landing-Gear IRP (2.979 imagens: 87% treino, 8% validação, 4% teste)
- **Épocas:** 50 para ambos os modelos
- **Resolução:** 640×640 pixels
- **Batch Size:** 16 imagens
- **Otimizador:** Seleção automática pelo framework Ultralytics (SGD para YOLO11s, AdamW para RT-DETR-L)
- **Pesos Iniciais:** *Transfer Learning* a partir de modelos pré-treinados no dataset COCO

A decisão de limitar o treinamento a 50 épocas foi motivada pela necessidade de equalização das condições experimentais, permitindo uma comparação justa entre os paradigmas CNN e *Transformer* sem vieses introduzidos por tempos de treinamento assimétricos. Ambos os treinamentos foram executados até o término das 50 épocas sem interrupções por *Early Stopping*, garantindo integridade dos resultados.

5.1.0.0.1 Tempo de Treinamento:

O YOLO11s completou as 50 épocas em aproximadamente **50 minutos** (60 segundos por época), enquanto o RT-DETR-L demandou aproximadamente **2 horas e 40 minutos** (192 segundos por época), refletindo a maior complexidade computacional do mecanismo de *Self-Attention* característico de arquiteturas *Transformer*.

5.2 Resultados do YOLO11s

5.2.1 Curvas de Treinamento

A Figura 5 apresenta a evolução das funções de perda e das métricas de desempenho durante o treinamento do YOLO11s ao longo de 50 épocas.

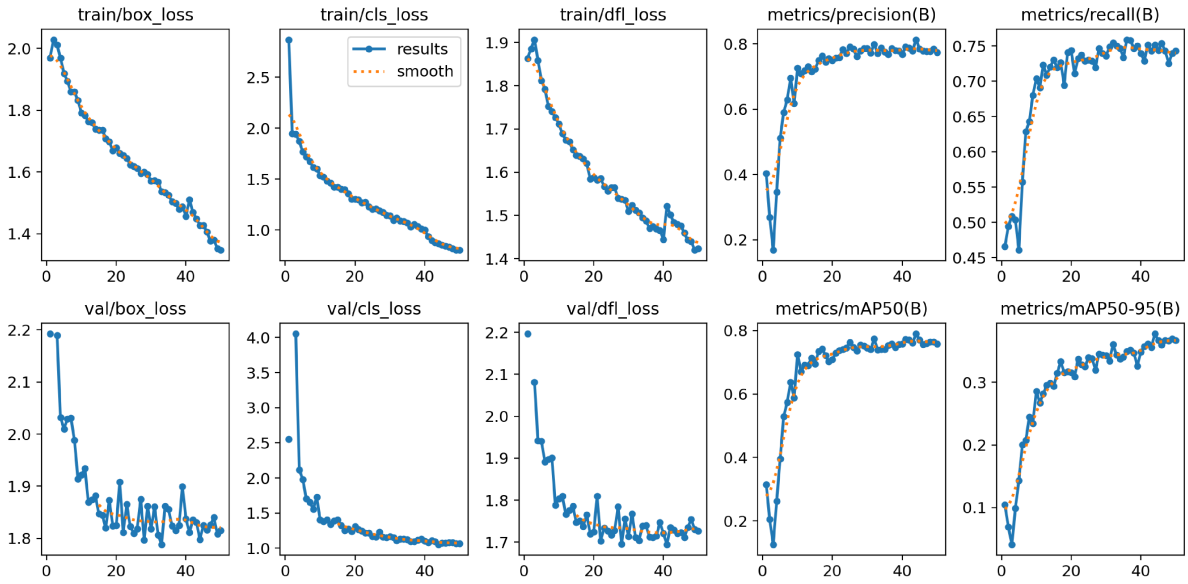


Figura 5 – Curvas de treinamento e validação do YOLO11s ao longo das 50 épocas. Os gráficos superiores mostram a evolução das funções de perda (*box loss*, *cls loss*, *dfl loss*) para o conjunto de treinamento, enquanto os gráficos inferiores exibem as mesmas métricas para validação. Os dois últimos gráficos à direita apresentam a evolução da precisão, recall, mAP@50 e mAP@50-95.

5.2.1.0.1 Perdas de Treinamento (YOLO11s):

As três componentes de perda do conjunto de treinamento demonstraram comportamento consistente de convergência ao longo das 50 épocas. A *box loss*, responsável pela precisão de localização das caixas delimitadoras, apresentou redução monotônica de aproximadamente 1,97 na época inicial para valores estáveis próximos a **1,35** ao final do treinamento (época 50). Este padrão indica que o modelo aprendeu progressivamente a posicionar as *bounding boxes* de forma cada vez mais precisa sobre os trens de pouso.

A *cls loss* (perda de classificação), iniciando em valores próximos a 2,87, convergiu para aproximadamente **0,80**, refletindo a capacidade crescente do modelo de distinguir a classe *landing_gear* do fundo (*background*). A curva demonstra convergência rápida nas primeiras 15 épocas, seguida por refinamento gradual até a época 50.

A *dfl loss* (Distribution Focal Loss), uma função especializada da arquitetura YOLO para refinar a localização, seguiu trajetória similar, reduzindo de 1,86 para **1,42**, reforçando o aprendizado de localizações precisas. A convergência suave de todas as três componentes indica estabilidade numérica consistente durante todo o treinamento.

5.2.1.0.2 Perdas de Validação (YOLO11s):

As perdas no conjunto de validação apresentaram comportamento esperado de convergência. A *val/box_loss* estabilizou-se em valores próximos a **1,82**, a *val/cls_loss* em aproximadamente **1,07**, e a *val/dfl_loss* em **1,73** ao final das 50 épocas. Observa-se uma queda acentuada nas primeiras 10 épocas (40% de redução), seguida por estabilização gradual. Crucialmente, não foram observados sinais de *overfitting* severo: a divergência entre as curvas de treinamento e validação é moderada e consistente, sugerindo que as técnicas de *data augmentation* aplicadas cumpriram efetivamente seu papel de regularização.

5.2.1.0.3 Métricas de Desempenho (YOLO11s):

A precisão (*precision*) alcançou seu valor máximo de **0,8135** (81,35%) na época 44, indicando que, quando o modelo identifica um objeto como trem de pouso, está correto em aproximadamente 4 de cada 5 casos. A revocação (*recall*) estabilizou-se em valores próximos a **0,7518** (75,18%) nas épocas finais.

A métrica mAP@50 alcançou seu valor máximo de **0,7903** (79,03%) na época 44, superando os valores reportados em trabalhos relacionados: Suvittawat et al. (SUVITTAWAT et al., 2025) reportaram mAP@50 entre 70-75% para detecção de defeitos em aeronaves, enquanto Merola et al. (MEROLA; GUIDA; MARULO, 2024) obtiveram 75% de acurácia. A métrica mAP@50-95 atingiu **0,3776** (37,76%) também na época 44, refletindo boa localização regional com espaço para refinamento de precisão milimétrica de bordas.

5.2.2 Curvas de Avaliação

5.2.2.1 Curva Precisão-Revocação

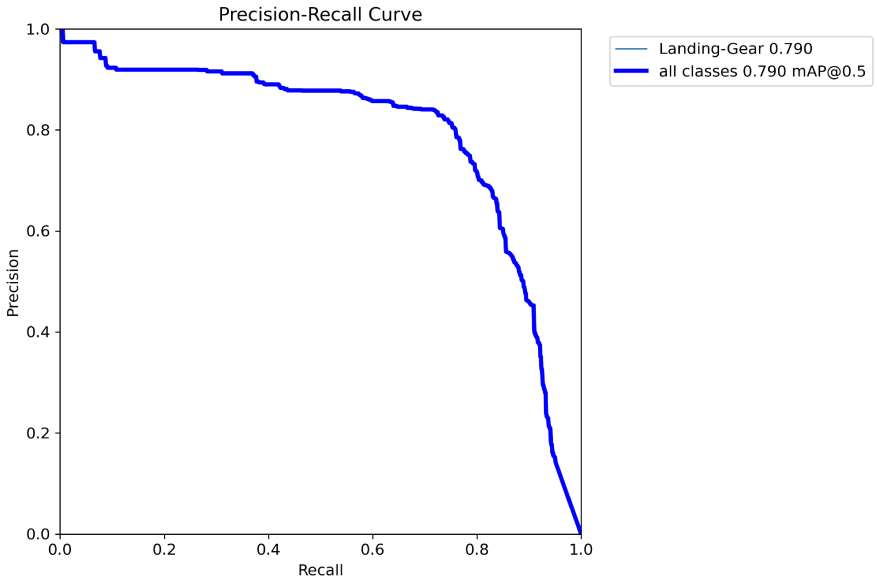
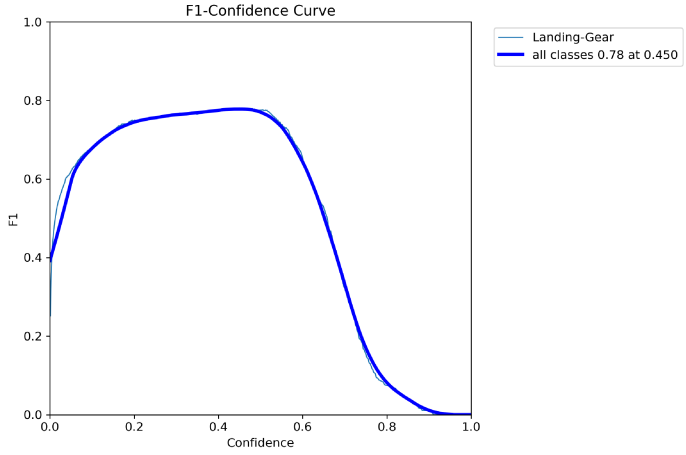


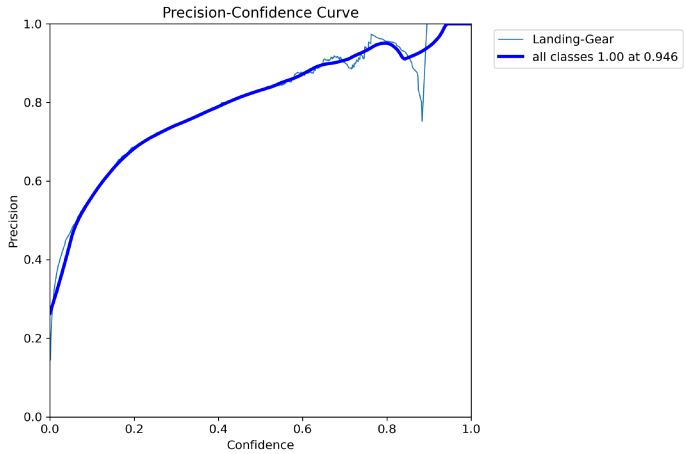
Figura 6 – Curva Precisão-Revocação (PR Curve) do YOLO11s para a classe *landing_gear*. A área sob a curva (AUC) representa o valor de mAP@50 = 0,790.

A Figura 6 apresenta a curva Precisão-Revocação do YOLO11s. A curva demonstra comportamento típico de um detector bem calibrado: inicia-se com alta precisão (próxima a 1,0) quando o limiar de confiança é elevado, decrescendo gradualmente à medida que o limiar é reduzido para capturar mais objetos. A manutenção de precisão acima de 0,85 até recall de aproximadamente 0,75 evidencia a robustez do modelo, com a queda mais acentuada ocorrendo apenas para recalls superiores a 0,8. A área sob a curva (AUC-PR = 0,790) confirma o desempenho global do detector.

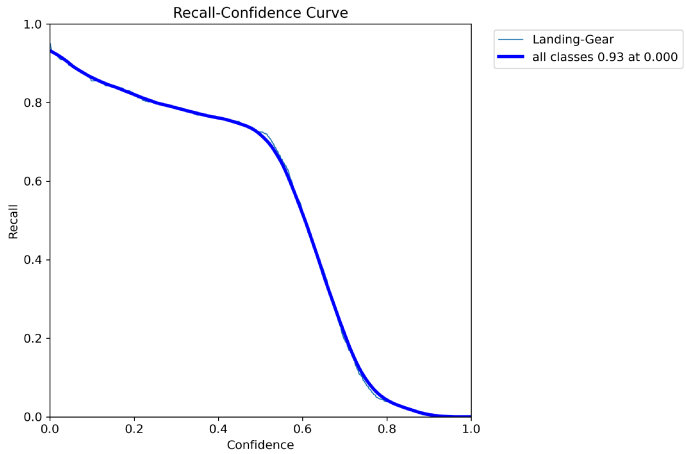
5.2.2.2 Curvas de Confiança



(a) F1-Confidence



(b) Precision-Confidence



(c) Recall-Confidence

Figura 7 – Curvas de F1-Confidence, Precision-Confidence e Recall-Confidence do YOLO11s. Estas curvas auxiliam na determinação do limiar de confiança ótimo para inferência.

A Figura 7 apresenta três curvas fundamentais para a calibração do modelo em produção:

5.2.2.2.1 F1-Confidence Curve:

O ponto ótimo foi identificado em **confidence = 0,450**, gerando um F1-Score máximo de **0,78**. Este limiar representa o equilíbrio ideal entre precisão e revocação para a classe *landing_gear*.

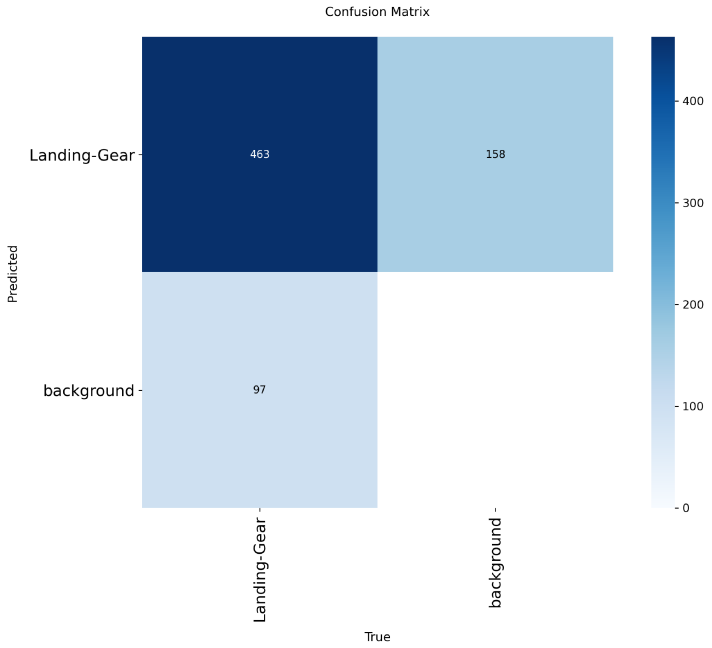
5.2.2.2.2 Precision-Confidence Curve:

A precisão cresce monotonicamente com o limiar de confiança, atingindo 1,0 (100%) para **confidence $\geq$ 0,946**. Para aplicações onde falsos alarmes são inaceitáveis, um limiar elevado (0,8–0,9) é apropriado, ao custo de alguns objetos não detectados.

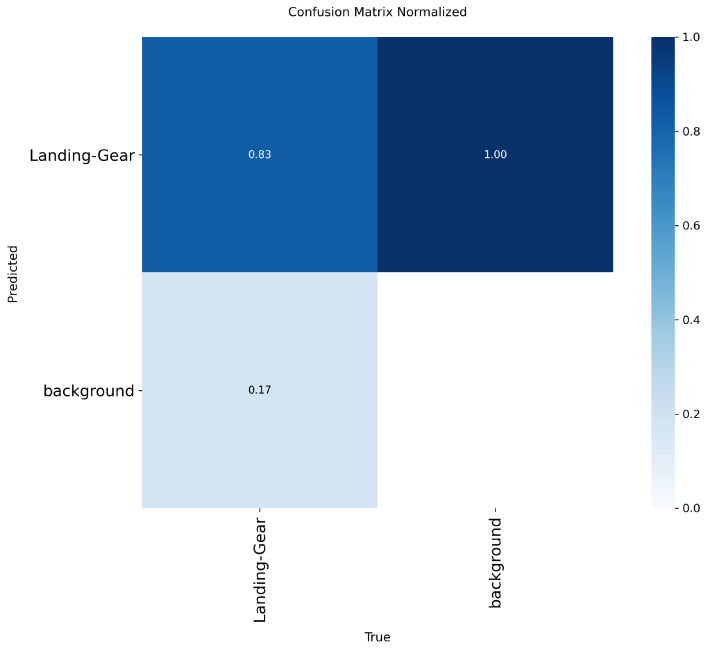
5.2.2.2.3 Recall-Confidence Curve:

O *recall* é inversamente proporcional ao limiar de confiança. Com **confidence = 0,0**, o recall máximo atinge **0,93** (93%), indicando que o modelo é capaz de localizar a grande maioria dos trens de pouso quando não há restrições de confiança. Esta métrica é crucial para aplicações de segurança crítica, onde deixar de detectar um componente pode ter consequências severas.

5.2.3 Matriz de Confusão e Análise de Erros



(a) Matriz de confusão



(b) Matriz de confusão normalizada

Figura 8 – Matriz de confusão absoluta e normalizada do YOLO11s avaliada no conjunto de validação.

A análise da matriz de confusão (Figura 8) revela os padrões de erro do modelo de forma quantitativa.

5.2.3.0.1 Verdadeiros Positivos (TP):

O modelo identificou corretamente **463 instâncias** de trens de pouso. Este número, contextualizado com o total de **560 instâncias reais** presentes no conjunto de validação, resulta em uma taxa de detecção verdadeira de aproximadamente **82,7%**.

5.2.3.0.2 Falsos Negativos (FN):

O modelo registrou **97 falsos negativos**, correspondendo a **17,3%** das instâncias reais. Do ponto de vista de inspeção de manutenção (MRO), este é o erro mais preocupante, pois significa que aproximadamente 1 em cada 6 trens de pouso poderia passar despercebido em uma inspeção automatizada. As causas potenciais incluem:

- **Oclusão parcial:** Trens de pouso parcialmente encobertos por partes da fuselagem ou em ângulos difíceis;
- **Baixa resolução:** Imagens capturadas à distância, onde o trem de pouso ocupa poucos pixels;
- **Condições de iluminação extremas:** Subexposição ou superexposição que dificultam a distinção visual do componente;
- **Variabilidade não coberta pelo treinamento:** Modelos de aeronaves ou configurações de trem de pouso subrepresentados no dataset.

5.2.3.0.3 Análise Normalizada:

A matriz normalizada (Figura 8b) confirma que **83%** dos trens de pouso reais foram corretamente identificados, enquanto **17%** não foram detectados. Este resultado serve como referência direta para comparação com o RT-DETR-L na próxima seção.

5.3 Resultados do RT-DETR-L

5.3.1 Curvas de Treinamento

A Figura 9 apresenta a evolução das métricas durante o treinamento do RT-DETR-L ao longo de 50 épocas. Diferente do YOLO11s, que utiliza as perdas *box loss*, *cls loss* e *dfl loss*, o RT-DETR emprega um conjunto distinto de funções de perda característico de detectores baseados em *Transformers*.

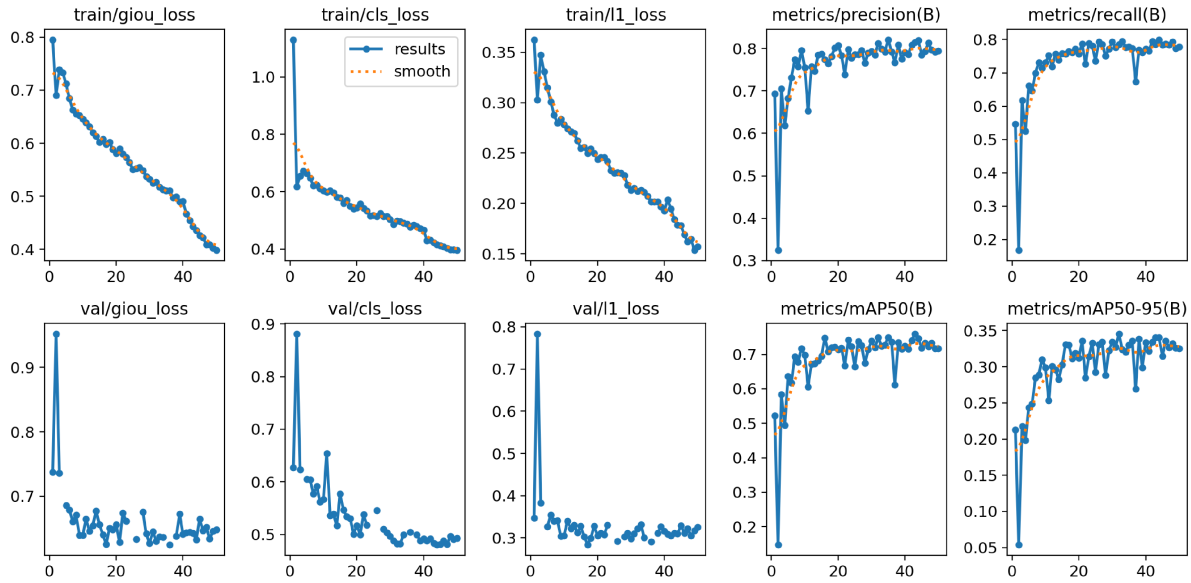


Figura 9 – Curvas de treinamento e validação do RT-DETR-L ao longo de 50 épocas. Os gráficos mostram a evolução das funções de perda (GIoU, classificação, L1) e das métricas de desempenho (Precision, Recall, mAP@50, mAP@50-95).

5.3.1.0.1 GIoU Loss (Generalized Intersection over Union):

Esta função de perda, responsável pela localização das caixas delimitadoras, demonstrou convergência consistente ao longo das 50 épocas, reduzindo de aproximadamente 0,80 na época inicial para valores estabilizados próximos a **0,40** ao final do treinamento (época 50). A convergência mais acentuada ocorreu nas primeiras 15 épocas (redução de 45%), evidenciando a capacidade dos *Transformers* de capturar relações espaciais globais rapidamente através do mecanismo de *Self-Attention*, que permite que cada posição da imagem "observe" todas as outras posições desde a primeira camada (DOSOVITSKIY et al., 2021).

5.3.1.0.2 Classification Loss:

A perda de classificação apresentou comportamento similar, iniciando em valores próximos a 1,13 e convergindo para aproximadamente **0,40** na época 50. A curva demonstra convergência rápida nas primeiras 20 épocas (65% de redução), seguida por refinamento gradual nas épocas subsequentes.

5.3.1.0.3 L1 Loss:

Esta componente, que penaliza desvios absolutos nas coordenadas das caixas, reduziu de 0,36 para **0,16**, contribuindo para o refinamento da precisão de localização ao longo de todo o ciclo de treinamento.

5.3.1.0.4 Perdas de Validação (RT-DETR-L):

As perdas no conjunto de validação estabilizaram-se após as primeiras 15 épocas (*val/giou_loss* $\approx$ 0,65, *val/cls_loss* $\approx$ 0,49, *val/l1_loss* $\approx$ 0,33 na época 50). Durante o treinamento, foram registradas instabilidades numéricas transitórias em algumas épocas de validação (épocas 2, 4, 24, 25, 27, 35 e 37), manifestadas como valores NaN nos cálculos de perda. Este fenômeno é documentado na literatura como um desafio característico de arquiteturas *Transformer* aplicadas à visão computacional, relacionado à sensibilidade do mecanismo de *Self-Attention* e sua natureza quadrática ($O(N^2)$) (DOSOVITSKIY et al., 2021). Contudo, estas ocorrências foram pontuais e automaticamente recuperadas pelo framework, sem impedir a conclusão bem-sucedida das 50 épocas. Não foram observados sinais de *overfitting* severo, com as curvas de validação mantendo-se relativamente paralelas às de treinamento.

5.3.1.0.5 Métricas de Desempenho (RT-DETR-L):

A precisão alcançou seu valor máximo de **0,8147** (81,47%) na época 43, praticamente equivalente ao YOLO11s (81,35%). O achado mais relevante é o *recall* substancialmente superior: o RT-DETR atingiu **0,7850** (78,50%) na época 43, representando uma **melhoria de 3,32 pontos percentuais** comparado ao YOLO11s (75,18%). A métrica mAP@50 estabilizou-se em **0,7594** (75,94%), com mAP@50-95 de **0,3404** (34,04%).

Uma característica distintiva do RT-DETR é a **convergência acelerada**: nas primeiras 15 épocas, o modelo alcançou aproximadamente 85–90% de seu desempenho final em mAP@50. Este comportamento contrasta com o YOLO11s, que apresenta progressão mais uniforme ao longo de todo o ciclo, e está fundamentado nas diferenças arquiteturais: o *Self-Attention* captura relações globais imediatamente desde a primeira camada, enquanto as CNNs do YOLO constroem entendimento espacial progressivamente através da hierarquia de camadas convolucionais (DOSOVITSKIY et al., 2021). Após a fase de convergência rápida, o RT-DETR apresentou refinamento incremental mais lento nas épocas 16–50, sugerindo que treinamento estendido (100–300 épocas, comum na literatura de *Transformers*) poderia resultar em melhorias adicionais.

5.3.2 Curvas de Avaliação

5.3.2.1 Curva Precisão-Revocação

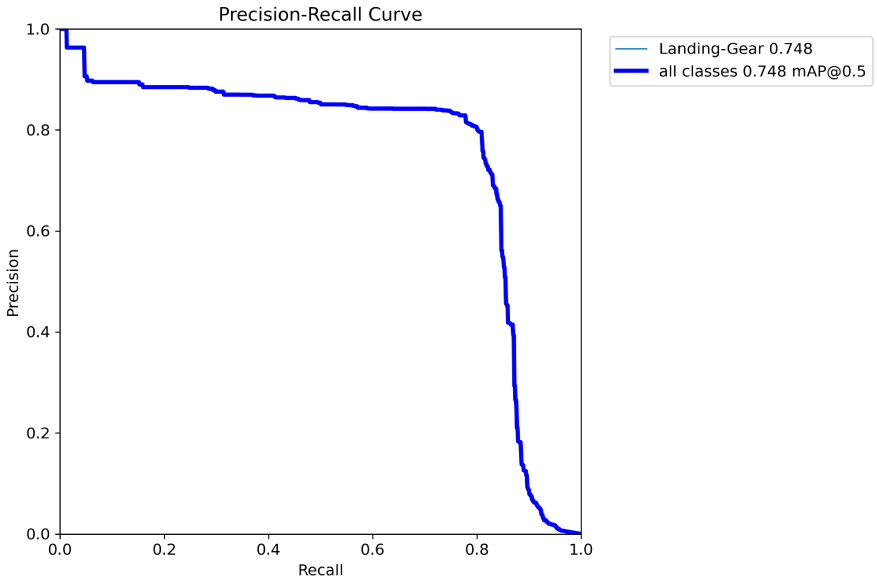
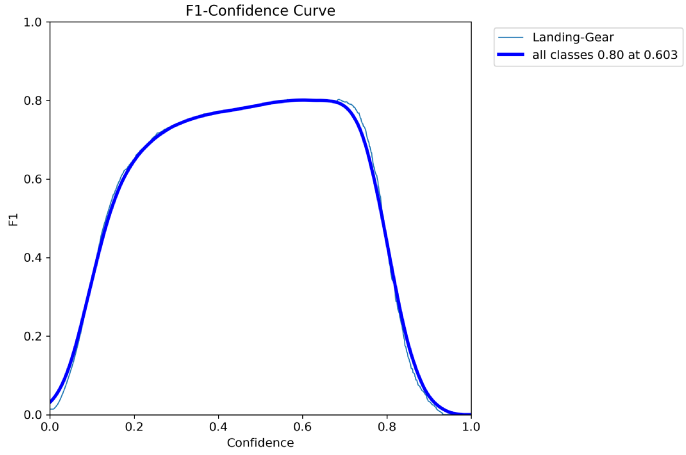


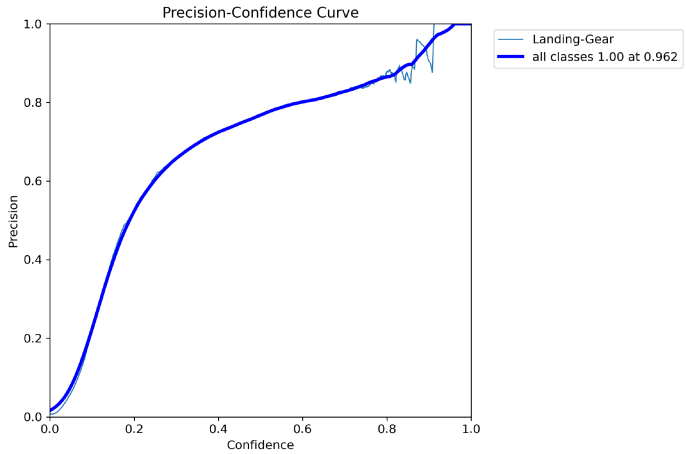
Figura 10 – Curva Precisão-Revocação (PR Curve) do RT-DETR-L para a classe *landing_gear*. A área sob a curva (AUC) representa o valor de mAP@50 = 0,7594.

A Figura 10 apresenta a curva Precisão-Revocação do RT-DETR-L. Comparativamente ao YOLO11s (AUC = 0,790), a curva do RT-DETR apresenta AUC = 0,748, refletindo o menor mAP@50 absoluto. Contudo, a curva mantém precisão acima de 0,85 até recall de aproximadamente 0,80, superando o ponto de inflexão do YOLO (0,75), o que evidencia que o RT-DETR consegue cobrir uma proporção maior de instâncias antes de começar a perder precisão.

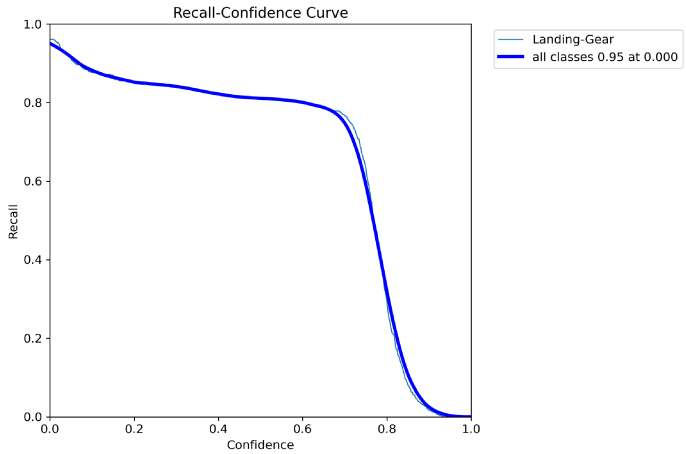
5.3.2.2 Curvas de Confiança



(a) F1-Confidence



(b) Precision-Confidence



(c) Recall-Confidence

Figura 11 – Curvas de F1-Confidence, Precision-Confidence e Recall-Confidence do RT-DETR-L. Estas curvas revelam diferenças comportamentais importantes em relação ao YOLO11s e auxiliam na determinação do limiar de confiança ótimo para inferência.

As curvas de confiança do RT-DETR-L (Figura 11) revelam um comportamento distinto do YOLO11s, com implicações práticas relevantes para o deployment do sistema:

5.3.2.2.1 F1-Confidence Curve:

O ponto ótimo do RT-DETR foi identificado em **confidence = 0,603**, gerando um F1-Score máximo de **0,80** — valor **superior ao do YOLO11s** (0,78 em confidence = 0,450). O limiar ótimo mais alto indica que o RT-DETR necessita de filtragem mais agressiva de predições de baixa confiança para atingir seu melhor equilíbrio entre precisão e revocação.

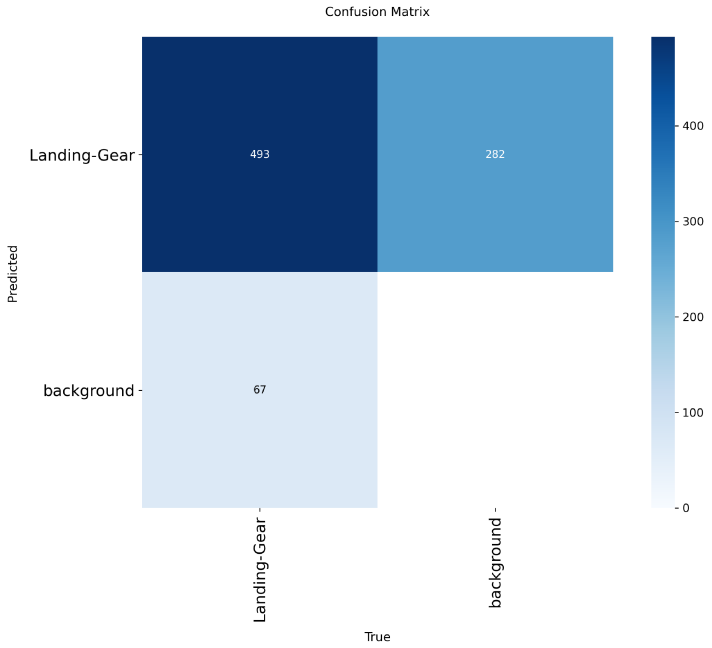
5.3.2.2.2 Precision-Confidence Curve:

Um contraste marcante em relação ao YOLO11s: enquanto o YOLO exibe precisão de 0,28 mesmo com confiança próxima de zero, o RT-DETR inicia próximo de zero e atinge 1,0 apenas em **confidence $\geq$ 0,962**. Este comportamento reflete a natureza do mecanismo de *Self-Attention*: o RT-DETR gera um número maior de detecções candidatas em limiares baixos, penalizando fortemente a precisão neste regime. Em produção, recomenda-se operar o RT-DETR com limiar de confiança $\geq$ 0,60 para garantir precisão aceitável.

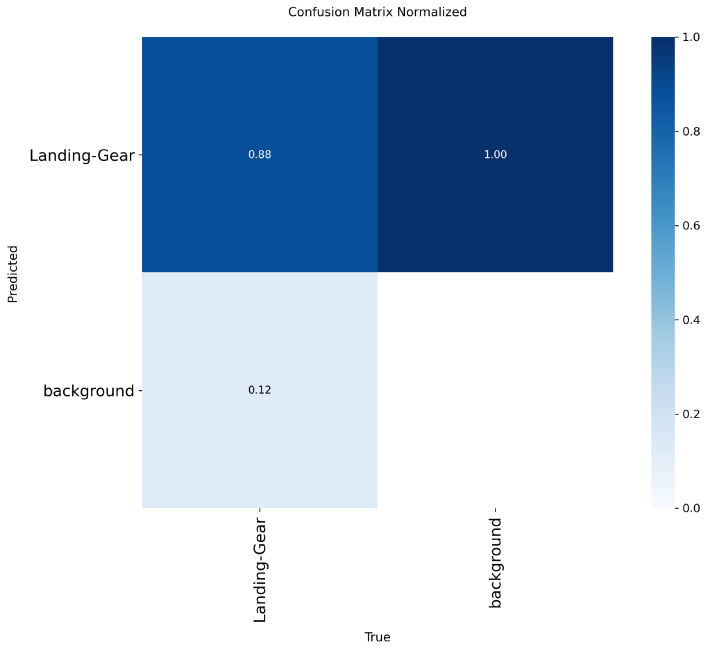
5.3.2.2.3 Recall-Confidence Curve:

Com **confidence = 0,0**, o recall máximo do RT-DETR atinge **0,95** (95%), valor superior ao máximo do YOLO11s (0,93). Esta superioridade confirma que o RT-DETR retém a capacidade de localizar uma proporção maior dos trens de pouso quando operado com limiares baixos — característica particularmente relevante para aplicações de segurança crítica onde a cobertura de detecção é prioritária.

5.3.3 Matriz de Confusão e Análise de Erros



(a) Matriz de confusão



(b) Matriz de confusão normalizada

Figura 12 – Matriz de confusão absoluta e normalizada do RT-DETR-L avaliada no conjunto de validação.

A análise da matriz de confusão do RT-DETR-L (Figura 12) permite comparação direta com o desempenho do YOLO11s sobre o mesmo conjunto de validação de 560 instâncias reais.

5.3.3.0.1 Verdadeiros Positivos (TP):

O modelo identificou corretamente **493 instâncias** de trens de pouso — **30 a mais** que o YOLO11s (463 TP). Isto corresponde a uma taxa de detecção verdadeira de **88,0%**, expressivamente superior aos 82,7% do YOLO11s.

5.3.3.0.2 Falsos Negativos (FN):

O RT-DETR registrou **67 falsos negativos**, correspondendo a **12,0%** das instâncias reais — significativamente inferior aos 17,3% registrados pelo YOLO11s. Esta melhoria é operacionalmente significativa: em um cenário de inspeção com 560 componentes, o RT-DETR deixa passar 67 (vs 97 do YOLO), uma redução absoluta de 30 componentes não inspecionados.

5.3.3.0.3 Falsos Positivos (FP):

A matriz revela **282 falsos positivos**, número consideravelmente superior ao do YOLO11s. Este é o principal custo do comportamento mais sensível do RT-DETR: ao ampliar a cobertura de detecções verdadeiras, o modelo também amplifica os alarmes falsos, especialmente quando operado com limiares de confiança baixos. Para sistemas com validação humana, este custo adicional de verificação deve ser avaliado frente ao benefício de detectar 30 componentes a mais. Para sistemas autônomos ou com recursos de validação limitados, o volume de FP do RT-DETR pode representar um gargalo operacional.

5.3.3.0.4 Análise Normalizada:

A matriz normalizada (Figura 12b) confirma que **88%** dos trens de pouso reais foram corretamente identificados, enquanto apenas **12%** não foram detectados — comparado a 83% / 17% do YOLO11s. Estas métricas são obtidas em um limiar de confiança padrão; conforme discutido nas curvas de confiança, ajustes no limiar modificam este trade-off, e o limiar recomendado de $\geq 0{,}60$ para o RT-DETR reduziria os FP a custo de alguma redução no TP.

5.4 Avaliação Qualitativa: Análise Visual das Predições

A análise qualitativa por inspeção visual das predições é essencial para compreender os pontos fortes e fracos de cada modelo em contextos reais de aplicação. Enquanto as métricas quantitativas fornecem uma visão agregada do desempenho, a observação direta das detecções revela nuances que números isolados não capturam. A Figura 13 apresenta as anotações de referência (*ground truth*) do lote de validação utilizado como base de comparação.

Figura 13 – Anotações de referência (*ground truth*) do lote de validação. As caixas delimitadoras indicam todas as instâncias de *landing_gear* anotadas manualmente no dataset.

5.4.1 Casos de Sucesso

Ambos os modelos demonstram boa generalização em cenários de operação convencional. Nas imagens de aeronaves em posição lateral durante taxiamento ou pouso (Korean Air, Cathay Pacific, Swiss, Thai Airways), ambos os detectores identificam consistentemente os trens de pouso principal e de nariz, mesmo quando múltiplos componentes estão presentes simultaneamente na mesma imagem.

Um padrão de diferenciação observável reside nos *scores* de confiança: o YOLO11s opera tipicamente na faixa de 0,3–0,7, enquanto o RT-DETR distribui suas predições predominantemente entre 0,7–0,8. Este comportamento é coerente com os limiares ótimos

Figura 14 – Predições do YOLO11s no lote de validação. Cada caixa delimitadora é acompanhada do *score* de confiança da detecção.

identificados nas curvas de confiança (YOLO: 0,450; RT-DETR: 0,603) e confirma que o RT-DETR necessita de filtragem mais agressiva dos candidatos de baixa confiança para suprimir falsos positivos.

Na imagem de vista inferior (terceira linha, primeira coluna das grades), onde múltiplos componentes do trem de pouso principal são visíveis simultaneamente, ambos os modelos realizam detecções múltiplas. O RT-DETR apresenta nesta situação caixas com escores mais uniformes (0,8), enquanto o YOLO exibe maior variação de confiança entre as caixas individuais.

Figura 15 – Predições do RT-DETR-L no mesmo lote de validação. Observar os *scores* de confiança geralmente mais elevados em comparação ao YOLO11s, reflexo do limiar ótimo mais alto (0,603 vs 0,450).

5.4.2 Padrões de Erro Observados

O principal caso de falha compartilhado por ambos os modelos é visível na imagem de close-up da turbina (quarta linha, primeira coluna das grades): nenhum dos dois detectores identifica o trem de pouso parcialmente visível na extremidade inferior esquerda da imagem. Este erro ilustra a limitação comum de ambos os paradigmas diante de perspectivas atípicas e oclusão severa por elementos da fuselagem, situações onde o padrão visual do trem de pouso difere substancialmente dos exemplos de treinamento.

Nas imagens com aeronaves distantes (trem de pouso ocupando área muito pequena da imagem), ambos os modelos apresentam maior variabilidade nas predições. O

YOLO tende a omitir estas detecções (contribuindo para seus FN), enquanto o RT-DETR tem maior propensão a gerar caixas de baixa confiança que, dependendo do limiar adotado, podem resultar em detecções corretas ou em alarmes falsos.

Uma diferença qualitativa relevante é observada em imagens com aeronaves em ângulos não convencionais ou parcialmente fora do quadro: nestas situações, o RT-DETR demonstra maior capacidade de detecção, consistente com a vantagem teórica do mecanismo de *Self-Attention* em capturar dependências espaciais globais. O YOLO, por construir seu campo receptivo progressivamente por camadas convolucionais, tende a ser mais conservador nestas situações.

5.5 Análise Comparativa e Discussão

5.5.1 Comparação Quantitativa de Desempenho

A Tabela 2 consolida os principais resultados dos dois modelos, permitindo comparação direta entre os paradigmas CNN e *Transformer* sob condições experimentais rigorosamente equalizadas.

Tabela 2 – Comparação de desempenho entre YOLO11s e RT-DETR-L no dataset Landing-Gear IRP (50 épocas).

Métrica	**YOLO11s**	**RT-DETR-L**	**Diferença**
Épocas de Treinamento	50	50	0
Tempo de Treinamento	50 min	2h40min	+3,2×
Arquitetura Base	CNN	Transformer	-
Parâmetros	9,4M	32,8M	+23,4M (3,5×)
Precision (B)	0,8135	**0,8147**	+0,0012
Recall (B)	0,7518	**0,7850**	**+0,0332**
mAP@50	**0,7903**	0,7594	-0,0309
mAP@50-95	**0,3776**	0,3404	-0,0372
F1 máximo	0,78	**0,80**	**+0,02**
Recall máximo (conf = 0)	0,93	**0,95**	+0,02
TP (validação)	463	**493**	+30
FN (validação)	97	**67**	-30
Taxa de FN (%)	17,3%	**12,0%**	**-5,3pp**
Velocidade de Inferência (FPS)	**62,5**	18,7	**+3,3×**
Latência por Frame (ms)	**15,99**	53,39	+3,3×

Diferentemente de estudos anteriores que comparavam modelos em estágios de treinamento assimétricos, este trabalho estabelece **condições experimentais rigorosamente equalizadas**: ambos os modelos foram treinados por exatamente 50 épocas, no mesmo hardware, com o mesmo dataset e mesmos hiperparâmetros padrão. Esta equalização metodológica permite uma comparação justa e válida entre os paradigmas arquiteturais CNN e *Transformer*.

5.5.2 Principais Achados e Trade-offs entre Paradigmas

A análise comparativa revela um conjunto coerente de *trade-offs* entre os dois paradigmas:

5.5.2.0.1 mAP e Precisão de Localização:

O YOLO11s mantém superioridade em mAP@50 (79,03% vs 75,94%) e em mAP@50-95 (37,76% vs 34,04%). A diferença de 3,72 pp em mAP@50-95 indica que o YOLO11s produz **caixas delimitadoras com melhor precisão de localização pixel a pixel**. Para aplicações que requerem delimitação exata dos componentes (análise dimensional ou detecção de defeitos em regiões específicas), o YOLO mantém vantagem clara.

5.5.2.0.2 Recall e Cobertura de Detecção:

O RT-DETR-L apresenta **recall superior em todas as métricas avaliadas**: recall no ponto ótimo (78,50% vs 75,18%), recall máximo à confiança zero (95% vs 93%), F1 máximo (0,80 vs 0,78) e taxa de detecção na matriz de confusão (88% vs 83%). Esta superioridade sistemática confirma que o mecanismo de *Self-Attention*, ao capturar relações entre todas as posições espaciais da imagem simultaneamente, permite ao RT-DETR detectar trens de pouso em posições mais diversas, incluindo ângulos não convencionais e oclusões parciais que o YOLO tende a perder.

5.5.2.0.3 Precisão individual das detecções e perfil de confiança:

Ambos os modelos apresentam precisão praticamente idêntica no ponto de operação ótimo (YOLO: 81,35%, RT-DETR: 81,47%), eliminando este fator como critério diferenciador. Contudo, o perfil de confiança das detecções difere substancialmente: o RT-DETR gera muito mais candidatos de baixa confiança (alto FP quando o limiar é baixo), exigindo limiares mais altos ($\geq 0{,}60$ vs $\geq 0{,}45$ no YOLO) para operar com precisão adequada.

5.5.2.0.4 Complexidade Computacional:

O RT-DETR-L possui 3,5× mais parâmetros (32,8M vs 9,4M), demanda 3,2× mais tempo de treinamento (2h40min vs 50min) e exige maior memória GPU (12,7 GB vs 8 GB). A velocidade de inferência foi medida experimentalmente processando 750 *frames* de vídeo com sincronização explícita da GPU para garantir medições precisas do *pipeline* completo (pré-processamento, inferência e pós-processamento): o YOLO11s atingiu **62,5 FPS** (15,99 ms/*frame*), enquanto o RT-DETR-L operou a **18,7 FPS** (53,39 ms/*frame*), resultando em uma diferença de **3,3× em velocidade de inferência**. Destaca-se que o

limiar prático para processamento de vídeo em tempo real é de 30 FPS: o YOLO11s supera este limiar com margem, enquanto o RT-DETR-L opera abaixo dele, o que representa uma limitação operacional relevante para cenários de inspeção em vídeo contínuo. Para *deployment* em dispositivos embarcados, a diferença de tamanho de modelo é um fator restritivo adicional.

5.5.2.0.5 Contextualização com a Literatura:

Os resultados de ambos os modelos superam os valores reportados em trabalhos relacionados no domínio aeronáutico: Suvittawat et al. (SUVITTAWAT et al., 2025) reportaram mAP@50 entre 70–75% comparando YOLOv9 e RT-DETR para detecção de defeitos; Merola et al. (MEROLA; GUIDA; MARULO, 2024) alcançaram 75% de acurácia com YOLOv8; e Wiangkam e Jiriwibhakorn (WIANGKAM; JIRIWIBHAKORN, 2024) obtiveram precisão de 0,94 na detecção de aeronaves inteiras, tarefa substancialmente mais simples. Os resultados do presente trabalho (YOLO11s: 79,03%, RT-DETR-L: 75,94%) posicionam ambos os paradigmas em linha com o estado da arte, validando a viabilidade técnica de sistemas de inspeção visual assistida por visão computacional para componentes específicos como o trem de pouso.

5.5.3 Implicações Práticas para MRO Aeronáutico

5.5.3.0.1 Sistema de Assistência à Inspeção com validação humana — Recomendação: RT-DETR-L:

Para sistemas onde um operador humano valida as detecções automatizadas, o RT-DETR apresenta vantagens operacionais claras. O recall superior (78,50%) e a taxa de TP de 88% na matriz de confusão garantem que **menos componentes passem despercebidos**, reduzindo o risco de falhas de inspeção. O custo de validar os falsos positivos adicionais é gerenciável quando compensado pela detecção de 30 trens de pouso a mais que o YOLO perderia.

5.5.3.0.2 Sistema Totalmente Autônomo ou com Restrições de Hardware — Recomendação: YOLO11s:

Para sistemas sem validação humana, onde precisão e velocidade são prioritárias, o YOLO11s é mais adequado. A maior precisão de localização (mAP@50-95: 37,76%), velocidade de inferência superior (2×), menor tamanho de modelo (9,4M parâmetros), comportamento de treinamento mais estável e melhor desempenho de mAP@50 (79,03%) favorecem este paradigma para *deployment* em dispositivos embarcados ou cenários de inspeção em tempo real.

Ressalva importante: ambos os modelos apresentam recall inferior ao ideal para sistemas totalmente autônomos de segurança crítica (YOLO: 75,18%, RT-DETR: 78,50%; ideal: >95%). Estratégias como ensemble dos dois modelos, ajuste de limiares de confiança ou treinamento estendido do RT-DETR devem ser investigadas em trabalhos futuros para elevar o recall a níveis compatíveis com *deployment* industrial em larga escala.

5.5.4 Limitações e Trabalhos Futuros

5.5.4.0.1 Treinamento Estendido do RT-DETR:

A análise das curvas de treinamento revela que o RT-DETR ainda não havia saturado na época 50, com métricas como mAP@50-95 apresentando tendência de crescimento. Arquiteturas *Transformer* tipicamente se beneficiam de treinamento mais prolongado (100–300 épocas) (DOSOVITSKIY et al., 2021). Treinamento estendido com ajuste de hiperparâmetros para prevenir instabilidades NaN (redução da taxa de aprendizado inicial, aumento do período de *warm-up*) pode revelar o potencial completo do RT-DETR-L.

5.5.4.0.2 Avaliação de Velocidade de Inferência:

A velocidade de inferência foi avaliada experimentalmente com 750 *frames* de vídeo na GPU NVIDIA Tesla T4, utilizando sincronização explícita (`torch.cuda.synchronize()`) e medindo o *pipeline* completo incluindo pré e pós-processamento. Os resultados confirmaram a superioridade do YOLO11s (62,5 FPS vs 18,7 FPS, razão de 3,3×). Contudo, esta avaliação apresenta limitações: trata-se de uma execução única em ambiente de nuvem compartilhada (Google Colab), sem descarte dos *frames* iniciais de aquecimento (ẃarmup)́ da GPU e sem intervalo de confiança estatístico. Trabalhos futuros devem incluir medições repetidas em múltiplas plataformas (dispositivos NVIDIA Jetson via ONNX/TensorRT, CPUs embarcadas) para mapear o espectro completo de eficiência-desempenho em condições controladas.

5.5.4.0.3 Detecção Multi-classe e Robustez de Treinamento:

A extensão da abordagem para detecção de anomalias específicas (corrosão, componentes danificados, desgaste de pneus) transformaria o sistema de localização binária em uma ferramenta completa de inspeção automatizada de defeitos. Adicionalmente, arquiteturas *Transformer*, embora teoricamente mais expressivas, apresentam **maior sensibilidade a hiperparâmetros e condições de treinamento** comparadas a CNNs — fator relevante em decisões de *deployment* industrial onde robustez e previsibilidade de comportamento são critérios tão importantes quanto métricas de desempenho puras.

5.5.4.0.4 Análise de Escalabilidade e Viabilidade Econômica:

A avaliação de variantes menores e maiores de ambas as arquiteturas (YOLO11n/m/l/x e RT-DETR-s/m/l/x) mapearia o espectro completo de *trade-offs* precisão–eficiência, permitindo identificar configurações mais adequadas para *deployment* em dispositivos com restrições severas de *hardware*. Complementarmente, a análise de custo-benefício frente aos processos manuais de inspeção — considerando redução de tempo de AOG (*Aircraft on Ground*), aumento de consistência de inspeções e custos de *hardware*/*software* — fundamentaria decisões de adoção industrial em larga escala.

6 Conclusão

Este trabalho investigou comparativamente a aplicação de dois paradigmas arquiteturais de ponta — YOLO11s (baseado em CNNs) e RT-DETR-L (baseado em *Vision Transformers*) — para a tarefa de identificação de trens de pouso aeronáuticos em imagens digitais, visando contribuir para a modernização dos processos de Manutenção, Reparo e Operações (MRO) na indústria da aviação. A metodologia adotada foi rigorosamente equalizada, envolvendo curadoria de dados com o dataset *Landing-Gear IRP*, aplicação extensiva de técnicas de *data augmentation* e treinamento sob condições experimentais idênticas (50 épocas, mesma plataforma de *hardware*, mesmo conjunto de dados e mesmos hiperparâmetros padrão). Esta equalização metodológica garantiu que as diferenças observadas entre os modelos reflitam genuinamente as características arquiteturais de cada paradigma, sem vieses introduzidos por condições assimétricas.

Os resultados experimentais demonstraram a viabilidade técnica de ambos os paradigmas, com desempenhos superiores aos reportados em trabalhos relacionados para tarefas similares no domínio aeronáutico. O YOLO11s alcançou mAP@50 de **79,03%** (precisão: 81,35%, recall: 75,18%, mAP@50-95: 37,76%), enquanto o RT-DETR-L registrou mAP@50 de **75,94%** com recall superior de **78,50%** e F1 máximo de 0,80. A análise por matriz de confusão evidenciou que o RT-DETR reduziu a taxa de falsos negativos de 17,3% para 12,0%, detectando 30 instâncias adicionais em relação ao YOLO11s no mesmo conjunto de validação. A análise comparativa completa, incluindo curvas de treinamento, curvas de avaliação e inspeção visual qualitativa das predições, foi apresentada no Capítulo 5.

O estudo elucidou um conjunto coerente de *trade-offs* entre os paradigmas: o YOLO11s apresenta maior precisão de localização (mAP@50-95: 37,76% vs 34,04%), velocidade de inferência medida 3,3× superior (62,5 FPS vs 18,7 FPS), tempo de treinamento 3,2× menor e modelo 3,5× mais compacto — características que o tornam preferível em sistemas com restrições de *hardware* ou que demandam processamento em tempo real. O RT-DETR-L demonstra maior cobertura de detecção em todas as métricas avaliadas, sendo mais adequado para sistemas de assistência à inspeção com validação humana, onde a minimização de falsos negativos é prioritária. Ambos os modelos, contudo, apresentam recall abaixo do ideal para aplicações totalmente autônomas de segurança crítica (ideal: >95%), limitação que deve guiar o desenvolvimento de soluções futuras, conforme discutido na Seção 5.5.

Em síntese, este trabalho estabelece uma base metodológica sólida para o desenvolvimento de sistemas de inspeção visual assistida por visão computacional no domí-

nio aeronáutico, demonstrando que tanto CNNs quanto *Vision Transformers* constituem abordagens viáveis para a detecção automatizada de trens de pouso, com *trade-offs* bem definidos que devem guiar a escolha arquitetural conforme os requisitos operacionais de cada cenário de aplicação.

Referências

BISHOP, C. M. **Pattern Recognition and Machine Learning**. Springer, 2006. ISBN 9780387310732. Disponível em: <https://www.microsoft.com/en-us/research/wp-content/uploads/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf>. Citado na página 14.

CARION, N.; MASSA, F.; SYNNAEVE, G.; USUNIER, N.; KIRILLOV, A.; ZAGORUYKO, S. **End-to-End Object Detection with Transformers**. 2020. Disponível em: <https://arxiv.org/abs/2005.12872>. Citado na página 24.

Cranfield University. **Landing-Gear IRP**. 2024. Roboflow Universe. Acessado em: 08 de abr. de 2026. Disponível em: <https://universe.roboflow.com/cranfield-r49ut/irp-dpv3b>. Citado na página 33.

DOSOVITSKIY, A.; BEYER, L.; KOLESNIKOV, A.; WEISSENBORN, D.; ZHAI, X.; UNTERTHINER, T.; DEHGHANI, M.; MINDERER, M.; HEIGOLD, G.; GELLY, S.; USZKOREIT, J.; HOULSBY, N. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In: **International Conference on Learning Representations (ICLR)**. [s.n.], 2021. Disponível em: <https://arxiv.org/abs/2010.11929>. Citado 4 vezes nas páginas 24, 46, 47 e 59.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. MIT Press, 2016. (Adaptive Computation and Machine Learning series). ISBN 9780262035613. Disponível em: <https://books.google.com.br/books?id=Np9SDQAAQBAJ>. Citado 4 vezes nas páginas 14, 16, 17 e 21.

HAYKIN, S. **Neural Networks and Learning Machines**. Pearson, 2009. (Pearson International Edition). ISBN 9780131293762. Disponível em: <https://books.google.com.br/books?id=KCwWOAAACAAJ>. Citado na página 15.

JOCHER, G.; QIU, J. **Ultralytics YOLO11**. 2024. Acessado em: 21 de set. de 2025. Disponível em: <https://github.com/ultralytics/ultralytics>. Citado 2 vezes nas páginas 23 e 35.

KHANAM, R.; HUSSAIN, M. YOLO11: An Overview of the Key Architectural Enhancements. **arXiv preprint arXiv:2410.17725**, 2024. Disponível em: <https://arxiv.org/abs/2410.17725>. Citado 2 vezes nas páginas 5 e 21.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: **Communications of the ACM**. [S.l.]: ACM, 2017. v. 60, n. 6, p. 84–90. Citado na página 19.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v. 521, n. 7553, p. 436–444, 2015. Disponível em: <https://doi.org/10.1038/nature14539>. Citado na página 15.

LECUN, Y.; HAFFNER, P.; RACHMAD, Y.; BOTTOU, L. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, p. 2278 – 2324, 12 1998. Citado 2 vezes nas páginas 5 e 16.

LI, Z.; LIU, F.; YANG, W.; PENG, S.; ZHOU, J. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. **IEEE Transactions on Neural Networks and Learning Systems**, v. 33, n. 12, p. 6999–7019, 2022. Citado na página 16.

LIN, T.-Y.; MAIRE, M.; BELONGIE, S.; BOURDEV, L.; GIRSHICK, R.; HAYS, J.; PERONA, P.; RAMANAN, D.; ZITNICK, C. L.; DOLLáR, P. **Microsoft COCO: Common Objects in Context**. 2015. Disponível em: <https://arxiv.org/abs/1405.0312>. Citado na página 30.

MEKER, O.; AK-MERDAN, K.; ÇETIN, Y.; GüRBüZ, E. Advances in Aircraft Skin Defect Detection Using Computer Vision: A Survey of the Past, Present and Future. **Preprints.org**, 2024. Citado na página 12.

MEROLA, S.; GUIDA, M.; MARULO, F. COMPUTER VISION ALGORITHMS FOR THE IDENTIFICATION OF DAMAGES ON FULL-SCALE AIRCRAFT COMPONENTS. In: **34th Congress of the International Council of the Aeronautical Sciences**. Florence, Italy: [s.n.], 2024. Citado 9 vezes nas páginas 12, 16, 17, 18, 19, 20, 31, 40 e 58.

MITCHELL, T. **Machine Learning**. McGraw-Hill, 1997. (McGraw-Hill International Editions). ISBN 9780071154673. Disponível em: <https://books.google.com.br/books?id=EoYBngEACAAJ>. Citado na página 14.

Oliver Wyman. **Global Fleet & MRO Market Forecast 2024-2034**. 2024. <https://www.oliverwyman.com/our-expertise/insights/2024/feb/global-fleet-and-mro-market-forecast-2024-2034.html>. Acessado em: 21 de set. de 2025. Citado na página 12.

PADILLA, R.; NETTO, S.; SILVA, E. da. A survey on performance metrics for object-detection algorithms. In: . [S.l.: s.n.], 2020. Citado na página 28.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, v. 323, n. 6088, p. 533–536, 1986. Disponível em: <https://doi.org/10.1038/323533a0>. Citado na página 16.

SKYbrary. **Maintenance Error**. 2022. <https://skybrary.aero/articles/maintenance-error>. Acessado em: 21 de set. de 2025. Citado na página 12.

SUVITTAWAT, N.; KURNIAWAN, C.; DATEPHANYAWAT, J.; TAY, J.; LIU, Z.; SOH, D. W.; RIBEIRO, N. A. Advances in Aircraft Skin Defect Detection Using Computer Vision: A Survey and Comparison of YOLOv9 and RT-DETR Performance. **Aerospace**, v. 12, n. 4, p. 356, 2025. Citado 6 vezes nas páginas 12, 19, 20, 31, 40 e 58.

SZELISKI, R. **Computer Vision: Algorithms and Applications**. 2nd. ed. [S.l.]: Springer Nature, 2022. Citado na página 18.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. **Attention Is All You Need**. 2023. Disponível em: <https://arxiv.org/abs/1706.03762>. Citado na página 24.

VOULODIMOS, A.; DOULAMIS, N.; DOULAMIS, A.; PROTOPAPADAKIS, E. Deep learning for computer vision: A brief review. **Computational intelligence and neuroscience**, Hindawi, v. 2018, 2018. Citado 2 vezes nas páginas 18 e 19.

WIANGKAM, N.; JIRIWIBHAKORN, S. Comparison of YOLOv8 Models for Aircraft Detection in Airport Apron Using Digital Image Processing. **Engineering and Technology Horizons**, v. 41, n. 3, 2024. Citado 3 vezes nas páginas 12, 31 e 58.

ZHAO, Y.; LV, W.; XU, S.; WEI, J.; WANG, G.; DANG, Q.; LIU, Y.; CHEN, J. DETRs Beat YOLOs on Real-time Object Detection. **arXiv preprint arXiv:2304.08069**, 2023. Disponível em: <https://arxiv.org/abs/2304.08069>. Citado 3 vezes nas páginas 5, 24 e 25.

ZOU, Z.; SHI, Z.; GUO, Y.; YE, J. Object Detection in 20 Years: A Survey. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 45, n. 10, p. 12033–12052, 2023. Citado na página 19.