Universidade Estadual do Paraná
Campus Apucarana

IGOR DA SILVA VIEIRA BENATO

DETECÇÃO E SEGMENTAÇÃO DE DANOS FÍSICOS EM MAÇÃS UTILIZANDO MASK R-CNN

APUCARANA–PR

2025

IGOR DA SILVA VIEIRA BENATO

DETECÇÃO E SEGMENTAÇÃO DE DANOS FÍSICOS EM MAÇÃS UTILIZANDO MASK R-CNN

Versão Preliminar de Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual do Paraná para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Jose Luis Seixas Junior

APUCARANA–PR

2025

Igor da Silva Vieira Benato
Detecção e Segmentação de Danos Físicos em Maçãs Utilizando Mask R-CNN/
Igor da Silva Vieira Benato. – Apucarana–PR, 2025-
62 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Jose Luis Seixas Junior

– Universidade Estadual do Paraná, 2025.

1. Palavra-chave1. 2. Palavra-chave2. I. Orientador. II. Universidade xxx. III. Faculdade de xxx. IV. Título

CDU 02:141:005.7

IGOR DA SILVA VIEIRA BENATO

DETECÇÃO E SEGMENTAÇÃO DE DANOS FÍSICOS EM MAÇÃS UTILIZANDO MASK R-CNN

Versão Preliminar de Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual do Paraná para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

Prof. Dr. Jose Luis Seixas Junior
Universidade Estadual do Paraná
Orientador

Prof. Dr. Guilherme Corredato Guerino
UNESPAR - Universidade Estadual do Paraná, campus Apucarana

Prof. Dr. Nicollas Mocelin Sdroievski
UNESPAR - Universidade Estadual do Paraná, campus Apucarana

Apucarana–PR, 04 de dezembro de 2025

Este trabalho é dedicado a...

A Deus, pela vida, força e sabedoria que tornaram este projeto possível.

À minha amada esposa, pelo apoio incondicional, paciência e motivação constante.

Aos meus pais e familiares, por serem o alicerce e a inspiração em minha jornada.

Aos amigos, pela valiosa convivência e por compartilharem a alegria das conquistas.

AGRADECIMENTOS

Agradeço profundamente a Deus, cuja presença me guiou e fortaleceu em cada etapa deste percurso.

À minha família, expresso meu amor e gratidão pelo incentivo constante que me motivou a seguir adiante mesmo diante das dificuldades.

Ao meu orientador, agradeço pela dedicação, pela paciência e pelas valiosas orientações que foram essenciais para a construção deste trabalho.

Aos professores que participaram da minha formação, deixo meu reconhecimento por todo o conhecimento compartilhado.

Aos amigos e colegas, obrigado pela parceria, pelas trocas e pelo apoio que tornaram esta caminhada mais leve e enriquecedora.

Por fim, agradeço a todos que, de alguma forma, contribuíram para esta conquista.

. **Detecção e Segmentação de Danos Físicos em Maçãs Utilizando Mask R-CNN**. 62 p. Trabalho de Conclusão de Curso – Versão Preliminar (Bacharelado em Ciência da Computação) – Universidade Estadual do Paraná, Apucarana–PR, 2025.

RESUMO

A inspeção de qualidade de maçãs é um processo fundamental para atender às normas regulatórias brasileiras (IN n.º 5/2006), porém a análise manual apresenta limitações de subjetividade e escalabilidade. Este trabalho avalia o desempenho e a utilidade prática de uma solução automatizada baseada em Visão Computacional, utilizando a arquitetura de rede neural Mask R-CNN para a segmentação de instâncias de danos físicos nas maçãs. A metodologia fundamentou-se na técnica de Transfer Learning com backbone ResNet-101, utilizando um dataset inicial de 300 imagens que foi ampliado para 1.800 amostras através de Data Augmentation. Foram realizados três experimentos comparativos, treinamento com congelamento de camadas (freezing), descongelamento total (unfreezing) e uso de dataset expandido. Os resultados demonstraram que as estratégias iniciais apresentaram dificuldades na tarefa de segmentação, refletidas na divergência da função de perda de validação. Ressalta-se que a desproporção entre o tamanho da imagem e o dano físico faz com que pequenas variações na máscara gerem penalizações elevadas no erro. Em contrapartida, o modelo treinado com o conjunto de dados expandido apresentou estabilidade e generalização superior. Conclui-se que a aplicação da Mask R-CNN, aliada ao aumento da variabilidade de dados, é uma abordagem eficaz, comprovando sua utilidade técnica para a mensuração precisa e segmentação de danos físicos em maçãs.

Palavras-chave: Visão Computacional. Mask R-CNN. Segmentação de instâncias.

. **Detection and Segmentation of Physical Damage in Apples Using Mask R-CNN**. 62 p. Final Project – Draft Version (Bachelor of Science in Computer Science) – State University of Paraná, Apucarana–PR, 2025.

ABSTRACT

Quality inspection of apples is a fundamental process for complying with Brazilian regulatory standards (IN No. 5/2006), however, manual analysis presents limitations in terms of subjectivity and scalability. This work evaluates the performance and practical utility of an automated solution based on Computer Vision, using the Mask R-CNN neural network architecture for segmenting instances of physical damage in apples. The methodology was based on the Transfer Learning technique with a ResNet-101 backbone, using an initial dataset of 300 images that was expanded to 1,800 samples through Data Augmentation. Three comparative experiments were performed: training with layer freezing, total unfreezing, and use of an expanded dataset. The results demonstrated that the initial strategies presented difficulties in the segmentation task, reflected in the divergence of the validation loss function. It is noteworthy that the disproportion between the image size and the physical damage causes small variations in the mask to generate high error penalties. In contrast, the model trained with the expanded dataset showed superior stability and generalization. It is concluded that the application of Mask R-CNN, combined with increased data variability, is an effective approach, proving its technical utility for the precise measurement and segmentation of physical damage in apples.

Keywords: Computer Vision. Mask R-CNN. Instance Segmentation.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

COCO	*Common Objects*
CNN	*Convolutional Neural Network*
FCN	*Fully Convolutional Network*
FPN	*Feature Pyramid Network*
IBGE	Instituto Brasileiro de Geografia e Estatística
IBICT	Instituto Brasileiro de Informação em Ciência e Tecnologia
LR	*Learning Rate*
MLP	*Multilayer Perceptron*
NBR	Norma Brasileira
R-CNN	*Region-based Convolutional Neural Network*
RGB	*Red, Green and Blue*
RNA	*Recurrent Neural Architecture*
ROI	*Region of Interest*
RPN	*Region Proposal Network*
SGD	*Stochastic Gradient Descent*
SVM	*Support Vector Machine*

SUMÁRIO

1 INTRODUÇÃO

A cultura da maçã movimenta mais de um bilhão de reais por ano no Brasil e gera milhares de postos de trabalho nas regiões Sul e Sudeste [1]. Para garantir a qualidade do fruto que chega ao consumidor, o Ministério da Agricultura, Pecuária e Abastecimento editou a Instrução Normativa n.º 5/2006, estabelecendo limites de cor, calibre e tolerância a defeitos para as categorias Extra, 1, 2 e 3 [2].

Para determinar a qualidade da fruta, são avaliados diversos aspectos, como coloração, características da epiderme (lisura, brilho, manchas) e lesões físicas. Muitas dessas características, especialmente as irregularidades na casca, dependem exclusivamente de avaliação visual.

Essa avaliação é geralmente realizada manualmente em processos de inspeção de qualidade, que dependem de avaliadores treinados para cumprir as diretrizes da instrução normativa n.º 5 [3]. Entretanto, a rapidez do processo e a fadiga visual geram erros e falta de padronização na classificação. Por isso, torna-se necessário apoiar ou substituir o trabalho manual por sistemas automatizados mais confiáveis.

Visto que a classificação regida pela normativa n.º 5/2006 [2] avalia quatro categorias de atributos relacionados a danos físicos, especificamente lesões cicatrizadas (leves e graves), danos mecânicos por impacto e lesões abertas, e considerando que a gravidade desses defeitos depende da mensuração da área ocupada na superfície da fruta (em milímetros ou centímetros quadrados), torna-se viável a automação dessa análise por meio de processamento computacional de imagens.

Nesse contexto, os avanços recentes em Visão Computacional, especialmente nas Redes Neurais Convolucionais (CNN), sistemas matemáticos capazes de extrair automaticamente padrões de cor e textura por meio de operações de convolução [4], oferecem uma alternativa promissora para a modernização da inspeção de frutas [5].

A aplicação desses modelos apresenta-se como uma solução eficiente para o controle de qualidade de maçãs. Considerando que os danos físicos na epiderme depreciam o produto final e possuem critérios de desclassificação baseados na extensão da área afetada, este trabalho utiliza a arquitetura Mask R-CNN. O objetivo central é avaliar o desempenho técnico e a utilidade prática do modelo na segmentação de instâncias dessas avarias, investigando sua capacidade de mensurar com precisão a extensão do dano.

Diferentemente de arquiteturas focadas exclusivamente na detecção de objetos (*bounding boxes*), o Mask R-CNN realiza a segmentação de instâncias, gerando máscaras que delimitam a região de interesse em nível de pixel. Tal capacidade é o que define a utilidade da ferramenta para a inspeção de maçãs, pois permite não apenas localizar, mas

também delimitar com exatidão a área de múltiplas avarias, fornecendo as métricas de área necessárias para mitigar a subjetividade humana.

Assim, a pesquisa visa avaliar o desempenho e a utilidade da arquitetura na segmentação de danos físicos em maçãs, investigando a capacidade do modelo em delinear com precisão a extensão das avarias. O trabalho justifica-se pelo potencial de modernização do setor, introduzindo uma abordagem automatizada capaz de mitigar a subjetividade humana e auxiliar na padronização do controle de qualidade de maçãs.

O trabalho começa apresentando a base teórica, explicando os conceitos de Redes Neurais Artificiais (RNA), desde a aplicação inicial com o Perceptron até a evolução para as Redes Neurais Convolucionais (CNNs). Logo depois, o texto explica o papel da "espinha dorsal" (*backbone*) na extração das características da imagem e detalha por que escolheu a arquitetura residual ResNet-101 para este estudo.

Ainda na parte teórica, o estudo destaca como a arquitetura Mask R-CNN evoluiu ao longo do tempo. O texto descreve os modelos que vieram antes (R-CNN, Fast R-CNN e Faster R-CNN) e mostra as melhorias que cada um trouxe até chegar na capacidade atual de fazer a segmentação de instâncias.

Na seção de trabalhos correlatos, a pesquisa analisa outros estudos que usaram a família R-CNN, principalmente o Mask R-CNN, na agricultura. Foram usadas referências sobre plantações de alface, quinoa e detecção de danos em maçãs para comparar resultados e definir as estratégias de melhoria do modelo.

O método de pesquisa define como as imagens foram coletadas e como o conjunto de dados (*dataset*) foi montado, além das configurações usadas no treinamento. Para medir os resultados, foram escolhidas métricas como a *Average Precision* (AP), *F1-Score* e *Intersection over Union* (IoU), além de acompanhar a função de perda (*Loss*) do próprio Mask R-CNN.

Nos experimentos, o trabalho detalha as três abordagens usadas para treinar a rede: a *Head* (treinando apenas as camadas de topo), a *All* (ajuste fino completo da rede) e a *Expanded* (usando técnicas para aumentar o número de imagens).

Por fim, nos resultados e conclusão, o texto avalia como o modelo se saiu em cada cenário. A conclusão é que, apesar de todas as estratégias mostrarem resultados interessantes, a abordagem do experimento *Expanded* teve o melhor desempenho, conseguindo generalizar melhor e ter mais precisão na segmentação.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Técnicas

2.1.1 Redes Neurais Artificiais

As Redes Neurais Artificiais (RNA) consistem em modelos computacionais inspirados na estrutura biológica e no funcionamento do cérebro humano, caracterizando-se por um conjunto de unidades de processamento simples (neurônios) interconectadas.

A unidade fundamental da RNA é o Neurônio Artificial, projetado para mimetizar a capacidade de aprendizado biológico. Um marco inicial dessa abordagem foi o modelo Perceptron, proposto por Rosenblatt [6]. Matematicamente, essa estrutura é composta por um vetor de entradas, pesos (coeficientes de ponderação), uma função de transferência (ou limiar) e uma função de ativação que determina a resposta de saída do neurônio. Podemos ver essa estrutura do Perceptron, composto por múltiplas entradas na Figura 1 [6].

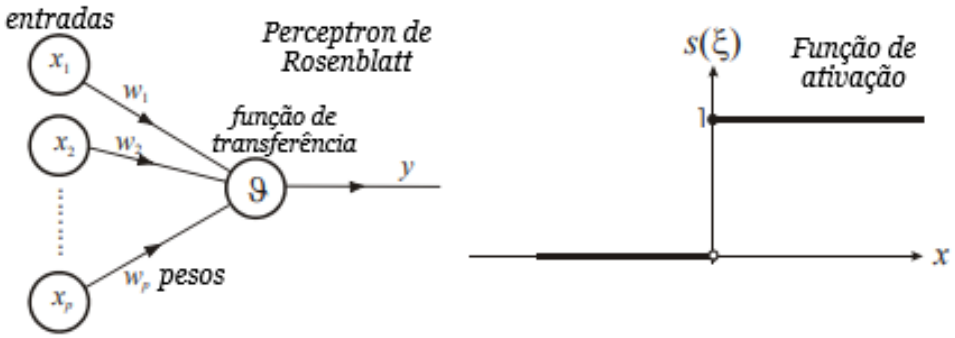

Figura 1 – Estrutura do Perceptron de Rosenblatt.

A camada de entrada recebe os dados brutos a serem processados pelo modelo. Cada entrada é associada a um peso, que atua como um coeficiente de ponderação. O ajuste desses pesos determina a relevância de cada dado de entrada, sendo esta a forma como a rede armazena o conhecimento extraído durante o treinamento.

Estes componentes, juntamente com a função de transferência (ou junção), permitem calcular o resultado da soma ponderada das entradas. Esse valor será posteriormente processado pela função de ativação, definindo a saída final da rede.

No Perceptron de Rosenblatt, por exemplo, utiliza-se a função degrau binária, a qual produz apenas dois valores de saída: 0 ou 1 (ou –1 e 1, em algumas variações). Essa resposta depende da comparação entre a soma ponderada e um valor fixo denominado

limiar (*threshold*): se a soma for maior ou igual ao limiar, a saída assume o valor 1; caso contrário, assume o valor 0.

As redes neurais são utilizadas para resolver problemas de tomada de decisão mais complexos, como a classificação, determinando se um conjunto de dados de entrada pertence ou não a uma classe. Entretanto, em sua forma inicial, o Perceptron apresentava limitações em tarefas que envolviam classificações não lineares, como demonstrado por Minsky [7].

Essas restrições foram superadas com o surgimento do Perceptron Multicamadas (MLP). Diferentemente do modelo simples, o MLP introduz uma ou mais camadas ocultas entre a entrada e a saída. Nessa estrutura, todas as saídas de uma camada se conectam com cada entrada da camada posterior, formando o que é conhecido como camadas totalmente conectadas (*fully connected*). Os sinais são propagados camada a camada, permitindo que a rede modele relações não lineares complexas que o Perceptron original não conseguia resolver.

Apesar de as MLPs superarem a limitação linear, elas apresentam sérias restrições ao processar dados de alta dimensionalidade, como imagens. Em arquiteturas totalmente conectadas, o número de parâmetros cresce exponencialmente com o aumento da profundidade e do tamanho da entrada, resultando em um custo computacional proibitivo e dificultando a generalização do modelo [8, 9].

2.1.2 Redes Neurais Convolucionais

As Redes Neurais Convolucionais (CNN ou *Convolutional Neural Networks*) são uma classe de RNAs utilizadas principalmente para o processamento de imagens, devido à sua capacidade de lidar com dados em formato de grade (matrizes). Essas redes destacam-se por utilizar operações de convolução para a extração de atributos, gerando os chamados mapas de características (*Feature Maps*) [4].

O processo de convolução resulta da aplicação de um filtro (ou *kernel*), que desliza sobre a matriz de entrada. Para cada posicionamento do filtro, é realizada a multiplicação elemento a elemento entre os pesos do filtro e a região correspondente da entrada, seguida da soma desses valores. O resultado de cada operação é armazenado em uma matriz resultante, denominada mapa de características (*Feature Map*), conforme apresentado na Figura 2 onde podemos ver um exemplo de convolução com filtro (*Kernel*) 3x3 [10].

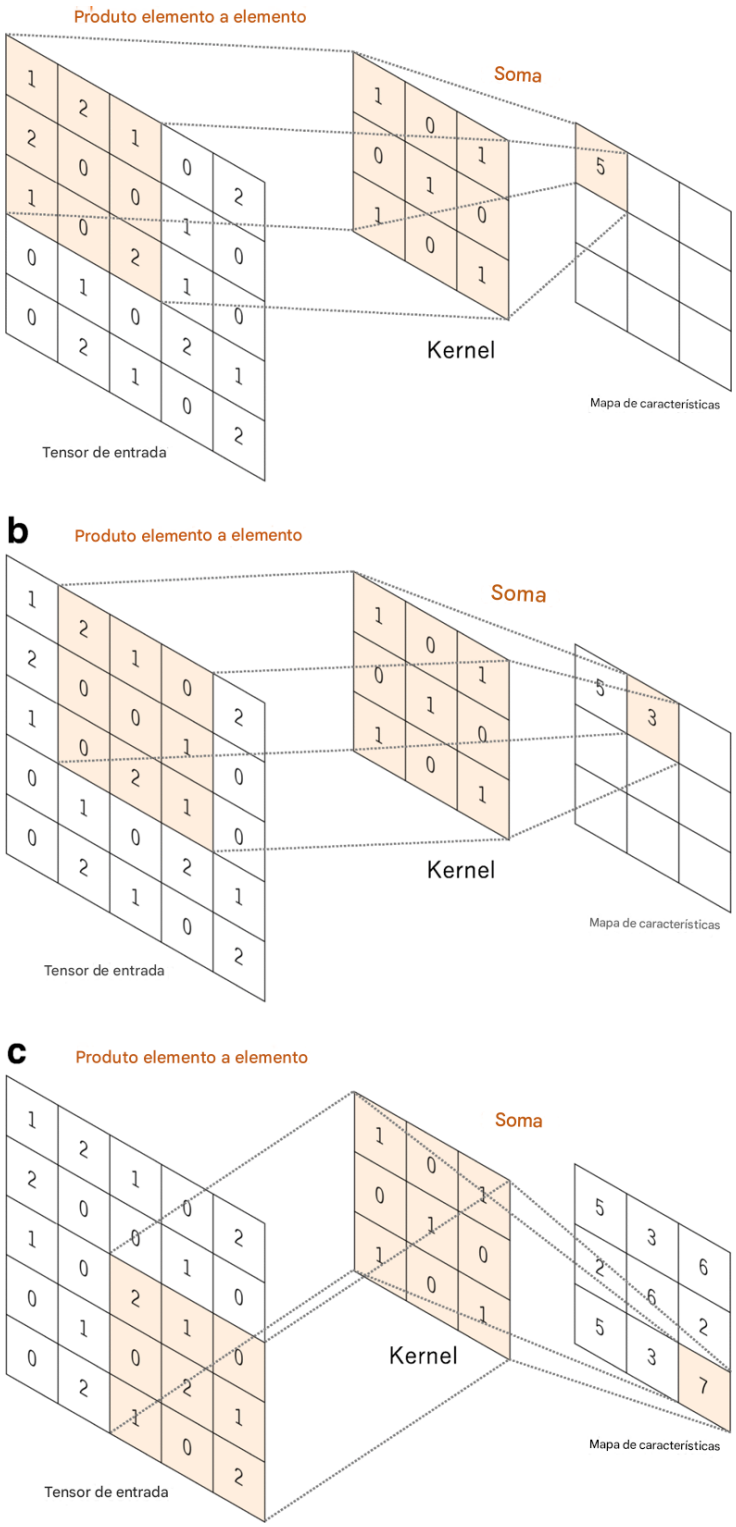

Figura 2 – Convolução com filtro.

As camadas que realizam esse processo são chamadas de camadas de convolução, principais responsáveis pela extração de características dos dados. Além da convolução, as CNNs também realizam outro processo importante para a extração das características chamado de *pooling*, que é o processo para abstrair os dados do mapa de características gerado pelas camadas de convolução.

Esse processo reduz a dimensão das entradas e resume os resultados do mapa de características. Um exemplo comum é a técnica de *max pooling*, que seleciona o maior valor resultante de uma região específica do mapa e gera uma nova saída de dimensão menor. Essa nova matriz é formada apenas pelos valores máximos de cada região, como demonstrado na Figura 3, onde podemos ver um exemplo de redução de dimensionalidade utilizando *max pooling* e *average pooling*.

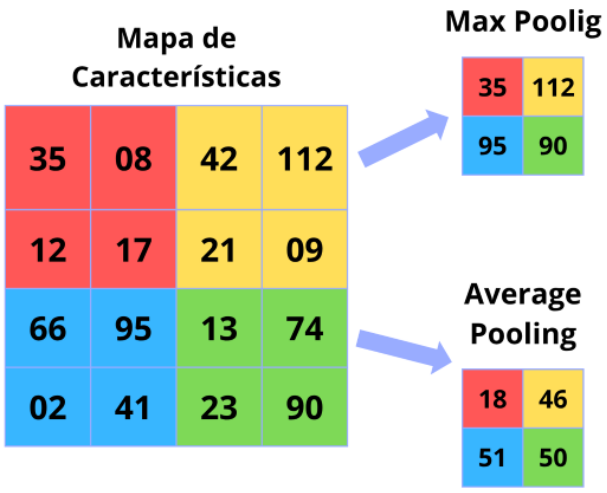

Figura 3 – Mapa de características.

Por fim, após a geração do mapa de características com o *pooling* aplicado, a matriz resultante é transformada em um vetor de uma única dimensão. Esse novo vetor é passado como entrada para as camadas de classificação [11], que geralmente utilizam camadas totalmente conectadas. Nessas camadas, a estrutura utilizada é igual às MLPs, onde cada entrada tem um respectivo peso e a saída de cada neurônio se conecta com a entrada de cada neurônio da camada posterior.

2.1.3 Arquiteturas Profundas e Resnet

Em arquiteturas modernas de detecção e segmentação de objetos, o processo de extração de características é delegado a uma CNN base, denominada *backbone* (espinha dorsal). O papel do *backbone* é transformar a imagem de entrada bruta em um conjunto rico de mapas de características (*feature maps*), que capturam desde detalhes de baixo nível (como bordas e cores) até estruturas semânticas complexas. A eficiência da detecção de danos em maçãs depende diretamente da capacidade do *backbone* de distinguir texturas sutis na epiderme da fruta.

Para aumentar a capacidade de representação de uma rede neural, a estratégia intuitiva é aumentar sua profundidade (número de camadas). Entretanto, He et al.[12]

demonstraram que o treinamento de redes excessivamente profundas enfrenta o problema da degradação: à medida que a profundidade aumenta, a precisão satura e começa a degradar rapidamente. Isso ocorre devido ao problema do desvanecimento do gradiente (*vanishing gradient*), onde o sinal de erro utilizado para ajustar os pesos torna-se extremamente pequeno ao propagar-se para as primeiras camadas, impedindo o aprendizado eficaz.

Para solucionar a degradação em redes profundas, He et al. [12] introduziram a arquitetura ResNet (*Residual Network*). A inovação central consiste na utilização de blocos residuais com conexões de atalho (*skip connections*). Diferentemente das redes tradicionais que tentam aprender uma função direta $H(x)$, os blocos residuais aprendem uma função residual $F(x)$, de modo que a saída do bloco seja dada pela Equação 2.1:

$$H(x) = F(x) + x \tag{2.1}$$

Essa conexão de identidade $(+x)$ permite que o gradiente flua livremente através da rede durante o processo de retropropagação, mitigando o problema do desvanecimento e permitindo o treinamento de redes com centenas de camadas.A operação $F(x) + x$ realiza o mapeamento de identidade, permitindo que a rede aprenda a função residual em vez da função original completa, além de utilizar a função de ativação *ReLU* (Unidade Linear Retificada) que basicamente transforma qualquer valor negativo em zero evitando o desaparecimento do gradiente[12].Podemos observar essa operação na Figura 4

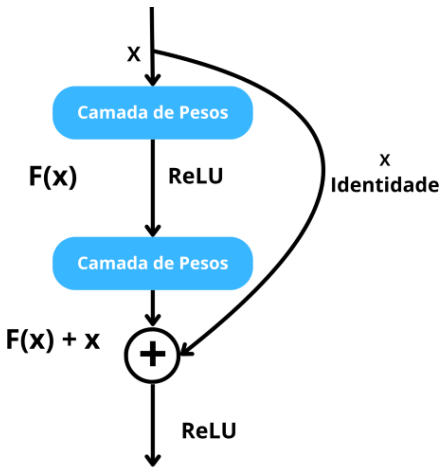

Figura 4 – Estrutura de um bloco de aprendizado residual.

Neste trabalho, utiliza-se a variante ResNet-101 como *backbone*. Esta arquitetura é composta por 101 camadas, estruturadas através de blocos do tipo *Bottleneck*, que utilizam convoluções 1×1 para reduzir a dimensionalidade antes das convoluções 3×3, otimizando o custo computacional.

O bloco *Bottleneck* opera manipulando a dimensão de profundidade (canais) para otimizar o processamento. O mecanismo divide-se em três etapas sequenciais: compressão, processamento e expansão. Inicialmente, ocorre a redução da dimensionalidade via convolução 1×1. Podemos ver a estrutura da operação na Figura 5, onde inicia com uma convolução 1×1 para reduzir a dimensionalidade (de 256 para 64 canais), seguida pela convolução espacial 3×3 e finalizando com a expansão para 256 canais, permitindo a conexão residual [12].

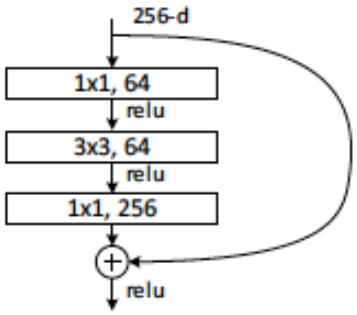

Figura 5 – Estrutura do bloco residual do tipo *Bottleneck*.

Exemplificando com uma entrada de $32 \times 32 \times 256$, a aplicação de 64 filtros 1×1 comprime a profundidade, gerando uma saída intermediária de $32 \times 32 \times 64$. Nesta etapa, cada filtro realiza uma combinação linear dos 256 canais de cada pixel. Em seguida, a convolução espacial (3×3) é executada sobre esse mapa reduzido, o que diminui drasticamente o custo computacional. Finalmente, ocorre a expansão, onde uma nova camada de convolução 1×1 com 256 filtros projeta os dados de volta à dimensão original, permitindo a soma com a entrada através da conexão residual [12].

2.1.4 Evolução do modelo R-CNN

As CNNs são empregadas principalmente para extrair características de imagens que, posteriormente, serão processadas por um classificador. No entanto, na tarefa de detecção de objetos, é necessário não apenas classificar, mas também localizar o objeto na imagem. Como uma única imagem pode conter múltiplos objetos em diferentes posições, torna-se necessária a definição de Regiões de Interesse (RoIs – *Regions of Interest*) para delimitar áreas com características relevantes.

A arquitetura R-CNN (*Regions with CNN features*), proposta por Girshick [13], foi uma das primeiras a integrar CNNs com propostas de regiões. Neste modelo, utiliza-se o algoritmo de Busca Seletiva (*Selective Search*) [14] para gerar milhares de propostas de regiões por imagem. Essas propostas são então recortadas e processadas individualmente por uma CNN para a extração de características.

Como a entrada da CNN possui dimensões fixas, as regiões propostas, originalmente de tamanhos variados, são submetidas a um processo de distorção (*warping*) para se adequarem à entrada da rede, conforme observado na Figura 6 [13]. Após essa etapa, a CNN extrai os vetores de características de cada região.

Para a etapa final de decisão, utiliza-se uma SVM (*Support Vector Machine*). Esta técnica consiste num algoritmo de aprendizado supervisionado que classifica os dados ao encontrar um hiperplano ótimo num espaço multidimensional, maximizando a margem de separação entre as diferentes classes [15]. Embora eficaz, a arquitetura R-CNN apresenta limitações de velocidade, servindo, no entanto, como uma das principais precursoras do modelo Mask R-CNN.

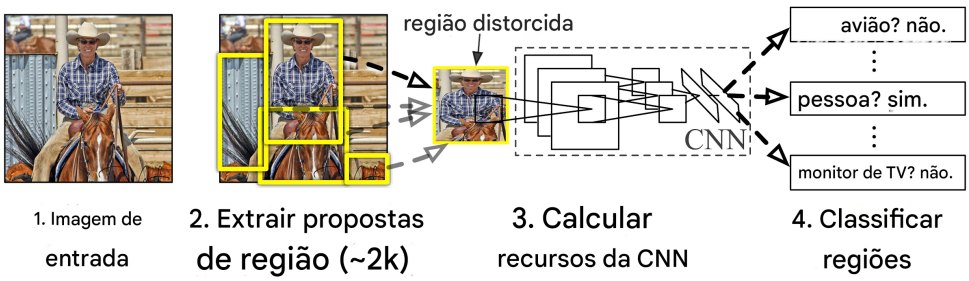

Figura 6 – Fluxo de processamento da arquitetura R-CNN.

2.1.5 Fast R-CNN

O Fast R-CNN apresentou mudanças significativas em sua arquitetura comparado ao seu antecessor, o R-CNN. A principal diferença é que a rede recebe como entrada a imagem inteira e o conjunto de RoIs simultaneamente. Diferentemente do R-CNN, onde a busca seletiva gerava milhares de regiões e cada uma passava individualmente pela CNN (gerando redundância), no Fast R-CNN a imagem é processada apenas uma vez.

No Fast R-CNN, a imagem original é processada integralmente pela CNN, gerando um mapa de características compartilhado. Sobre esse mapa, aplica-se a camada de RoI *Pooling*, proposta por Girshick [16], que visa transformar a área de cada RoI (que possui tamanho variável) em uma dimensão de tamanho fixo ($H \times W$). Essa técnica divide a região de interesse em uma grade de sub-regiões e aplica o *pooling* máximo em cada uma delas. Desta forma, extrai-se o valor mais relevante de cada seção, resultando em uma matriz de tamanho fixo, independentemente da dimensão original da proposta.

Por fim, a rede gera duas camadas de saída, uma responsável por realizar a classificação do objeto na imagem usando a função *softmax* e a outra responsável pela regressão da *bounding box*, que indica onde o objeto se encontra na imagem, conforme ilustrado na Figura 7.

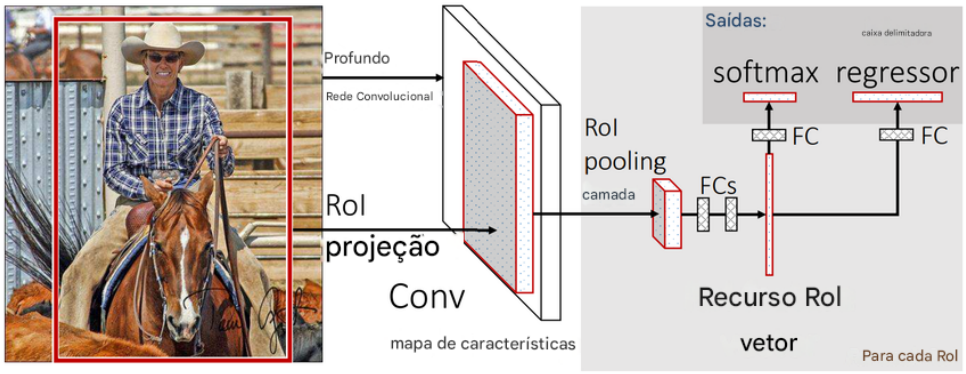

Figura 7 – Arquitetura do Fast R-CNN.

2.1.6 Faster R-CNN

Após os avanços introduzidos pelo Fast R-CNN, um novo modelo foi proposto com o objetivo de otimizar o processo de geração de regiões de interesse: o Faster R-CNN. Nesse modelo, foi apresentada a RPN (*Region Proposal Network*), que substituiu o uso da busca seletiva [14] na etapa de propostas de região. Em vez de depender de um método externo e não treinável, adicionou-se à arquitetura uma rede totalmente integrada, capaz de gerar propostas diretamente a partir dos mapas de características obtidos pelas primeiras camadas convolucionais.

A introdução da RPN impactou significativamente o desempenho do modelo. Anteriormente, a busca seletiva representava um gargalo computacional por ser executada na CPU e não utilizar os recursos de aprendizado da rede. Com a RPN integrada e utilizando os dados do mapa de características compartilhado (na GPU), a geração das propostas tornou-se muito mais eficiente, conforme demonstrado por Ren et al.[17].

Para viabilizar a detecção de objetos com diferentes escalas e proporções, a RPN introduz o conceito de âncoras (*anchors*). Em vez de tentar prever as coordenadas de uma região do zero, a rede utiliza um mecanismo de janela deslizante sobre o mapa de características. Para cada posição dessa janela, são geradas múltiplas caixas de referência (as âncoras) com diferentes escalas e razões de aspecto pré-definidas.

Conforme proposto por Ren et al. [17], utilizam-se tipicamente 3 escalas e 3 razões de aspecto (ex: 1:1, 1:2 e 2:1), totalizando $k = 9$ âncoras para cada ponto do mapa de características, conforme demonstrado na Figura 8.

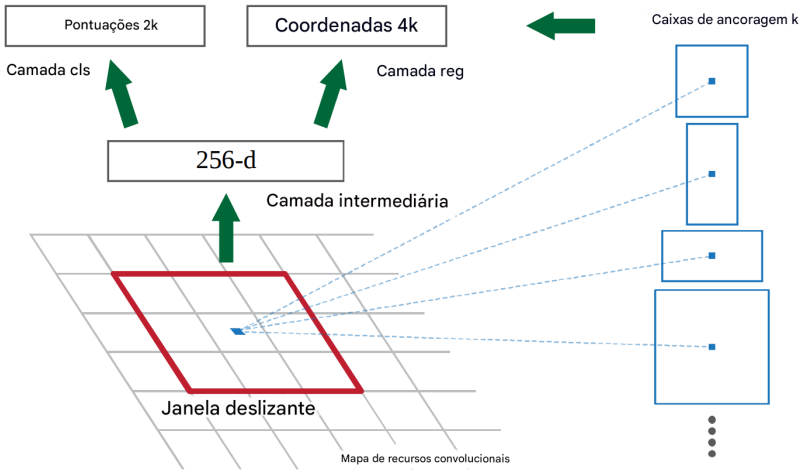

Figura 8 – Funcionamento da Rede de Proposta de Região (RPN).

Para cada âncora, a RPN prediz simultaneamente dois valores: a probabilidade de a caixa conter um objeto e os quatro deslocamentos (*offsets*) necessários para ajustar as coordenadas da âncora à posição real do objeto (*bounding box regression*).

Para treinar a rede a realizar as tarefas de classificação e regressão de caixas simultaneamente, Ren et al.[17] definiram uma função de perda multitarefa (*multi-task loss*). O objetivo é minimizar o erro tanto na identificação da presença do objeto quanto na precisão geométrica da caixa. A função de perda global para uma imagem é definida pela Equação 2.2:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{2.2}$$

Neste contexto, i é o índice de uma âncora no *minibatch* e p_i é a probabilidade prevista de a âncora ser um objeto. O termo p_i^* representa o rótulo verdadeiro (*ground truth*), sendo 1 se a âncora for positiva (contém objeto) e 0 se for negativa (fundo).

O primeiro componente, $\mathbf{L_{cls}}$, representa a perda de classificação (referente à probabilidade de *objectness*). Trata-se de uma função de perda logarítmica (*log loss*) sobre duas classes (objeto vs. não-objeto), ensinando a rede a distinguir quais âncoras estão sobrepondo os objetos reais.

O segundo componente, $\mathbf{L_{reg}}$, refere-se à perda de regressão (saída *box*). É fundamental notar que este termo é multiplicado por p_i^*, o que significa que a regressão da caixa só é ativada e contabilizada quando há, de fato, um objeto na âncora ($p_i^* = 1$). Para o cálculo desta perda, utiliza-se a função *Smooth L1* definida por Girshick et al. [16], que é mais robusta a *outliers* do que a perda L2 tradicional (erro quadrático).

O cálculo geométrico realizado pela camada de regressão não prevê as coordenadas absolutas, mas sim parâmetros de transformação parametrizados. Sendo (x, y, w, h) as

coordenadas do centro, largura e altura da caixa prevista, e (x_a, y_a, w_a, h_a) as da âncora, a rede aprende os seguintes deslocamentos:

- Deslocamento de Centro:

$$\Delta x = \frac{x - x_a}{w_a} \quad \text{e} \quad \Delta y = \frac{y - y_a}{h_a}$$

- Deslocamento de Escala:

$$\Delta w = \log\left(\frac{w}{w_a}\right) \quad \text{e} \quad \Delta h = \log\left(\frac{h}{h_a}\right)$$

Dessa forma, o modelo aprende a ajustar a âncora fixa para que ela se encaixe perfeitamente no objeto alvo, garantindo invariância à escala e translação.

2.1.7 Mask R-CNN

O *framework* Mask R-CNN, proposto por He et al.[18], é uma extensão direta do Faster R-CNN. Enquanto seu antecessor possui duas saídas principais para cada objeto candidato (uma etiqueta de classe e uma *bounding box*) conforme demosntrado na Figura 9, o Mask R-CNN adiciona um terceiro ramo paralelo responsável por prever a máscara de segmentação do objeto (máscara binária).

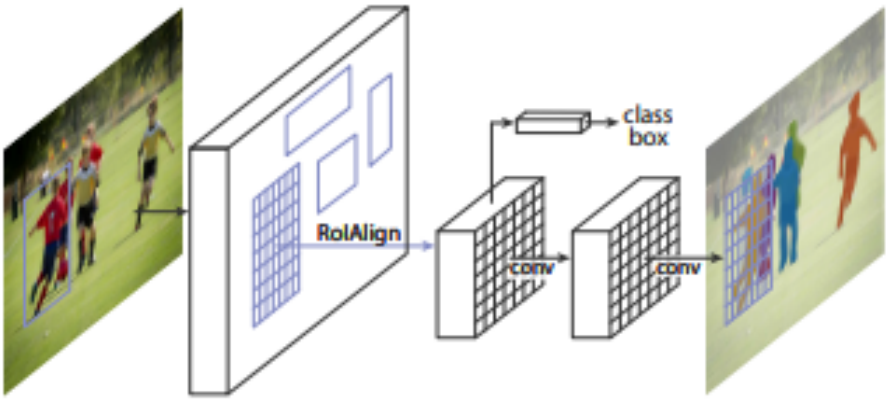

Figura 9 – Estrutura da arquitetura Mask R-CNN.

A grande inovação desta arquitetura é o desacoplamento da segmentação em relação à classificação. O ramo da máscara gera uma matriz $m \times m$ para cada classe possível, aplicando uma função sigmoide em cada pixel, sem competir com a predição de classes. Isso simplifica o fluxo de processamento do modelo e permite que o treinamento ocorra de ponta a ponta. A função de perda multitarefa é definida pela Equação 2.3:

$$L = L_{cls} + L_{box} + L_{mask} \tag{2.3}$$

Onde L_{cls} e L_{box} são idênticos aos definidos no Fast R-CNN (classificação e regressão da caixa), e L_{mask} é a entropia cruzada binária média, calculada pixel a pixel, considerando apenas a máscara associada à classe verdadeira (*ground truth*) [18].

Para gerar essas máscaras preservando o arranjo espacial dos pixels, o Mask R-CNN utiliza uma Rede Totalmente Convolucional (FCN) [19] aplicada a cada Região de Interesse. No entanto, para que isso fosse possível com precisão, foi necessário corrigir um problema estrutural das arquiteturas anteriores: a quantização.

Os modelos Fast e Faster R-CNN utilizavam a camada de *RoI Pooling*, que realizava arredondamentos (quantização) nas coordenadas flutuantes para ajustá-las à grade de características. Embora inofensivos para a detecção de caixas, esses arredondamentos causavam um desalinhamento pixel a pixel, prejudicando a segmentação. Para solucionar isso, [18] introduziram a camada RoI *Align*.

Diferentemente do *RoI Pooling*, o *RoI Align* elimina a quantização das coordenadas. A técnica utiliza interpolação bilinear para calcular os valores exatos das características em quatro pontos de amostragem dentro de cada célula da RoI, garantindo um alinhamento espacial preciso entre a entrada e a máscara de saída, fator crucial para a detecção de avarias pequenas ou irregulares na superfície da maçã.

2.2 Trabalhos Correlatos

A aplicação de Visão Computacional na agricultura tem passado por uma transformação significativa nas últimas décadas. Historicamente, a inspeção de qualidade dependia de processos manuais ou de técnicas clássicas de processamento de imagens, que utilizavam filtros de cor e limiares fixos. No entanto, essas abordagens mostravam-se limitadas em ambientes não controlados ou diante de defeitos com baixa variação cromática. Recentemente, o estado da arte migrou para soluções baseadas em Aprendizado Profundo (*Deep Learning*), especificamente Redes Neurais Convolucionais, devido à sua robustez e capacidade de generalização na detecção de padrões complexos em frutas e vegetais.

Neste cenário, diversas arquiteturas têm sido propostas para a detecção de avarias. Uma das abordagens mais relevantes recentemente foi apresentada por Hou et al. [20]. Neste estudo, os autores utilizaram o modelo Faster R-CNN para identificar machucados recentes e sutis em maçãs. Vale ressaltar que, para viabilizar a detecção dessas lesões de baixa percepção a olho nu, os autores recorreram a imagens hiperespectrais, tecnologia que realça as regiões danificadas em faixas espectrais específicas.

O uso dessa tecnologia evidencia o grau de desafio técnico deste trabalho, que propõe detectar avarias utilizando apenas imagens convencionais (RGB). Para compensar a ausência de dados espectrais e conseguir diferenciar danos sutis de manchas naturais, esta pesquisa avança ao adotar o *backbone* ResNet-101, uma rede mais profunda e capaz

de extrair características semânticas mais complexas do que a ResNet-50 utilizada por Hou et al. [20].

Além da questão do tipo de imagem usada, há uma diferença fundamental na arquitetura de saída. Embora o método de Hou et al. [20] tenha sido eficaz na localização, a saída limita-se a *bounding boxes*. Essa característica representa uma restrição técnica significativa, pois a caixa engloba tanto a área danificada quanto parte íntegra da casca.

Nesse contexto de uso de imagens convencionais, destaca-se o trabalho de El Akrouchi et al. [21], que aplicaram o Mask R-CNN para a detecção e segmentação de panículas de quinoa. O desafio central do estudo residiu na complexidade visual do ambiente de campo, caracterizado pela baixa distinção entre o objeto e a vegetação de fundo.

Além de validarem o uso de imagens RGB, os autores conduziram uma avaliação comparativa crucial utilizando três *backbones* distintos: ResNet-50, ResNet-101 e EfficientNet-B7. Essa análise demonstrou empiricamente como a variação na arquitetura e na profundidade da rede influencia a capacidade de aprendizado e o desempenho final da detecção. Para a presente pesquisa, esses resultados são relevantes, pois corroboram a importância de selecionar um *backbone* robusto (como a ResNet-101) para lidar com tarefas de detecção em cenários onde as características visuais são sutis ou complexas.

Um dos principais desafios no treinamento de redes neurais é a escassez de dados anotados, especialmente em cenários onde as características a serem analisadas são sutis. Contudo, conforme demonstrado por Osorio et al. [22], essa limitação pode ser mitigada através de técnicas como *transfer learning* e estratégias de anotação progressiva.

Em seu trabalho, os autores aplicaram o modelo Mask R-CNN para a detecção e contagem de culturas de alface e batata. O estudo utilizou um *dataset* reduzido, dividido em duas categorias de anotação: um conjunto com demarcações simples e outro com delimitações refinadas.

Os experimentos iniciais utilizaram os pesos pré-treinados do *dataset* COCO [23]. Ao treinar o modelo apenas com as anotações simples, os resultados mostraram-se insatisfatórios. Em contrapartida, o uso direto das anotações refinadas proporcionou uma melhora significativa no desempenho.

Além disso, Osorio et al. [22] exploraram uma estratégia de *transfer learning* em dois estágios. Inicialmente, a rede foi treinada com o *dataset* de anotações simples, partindo dos pesos do COCO. Subsequentemente, utilizaram-se os pesos resultantes dessa etapa para iniciar um novo ciclo de aprendizado com as anotações refinadas. Essa abordagem superou o desempenho do modelo treinado exclusivamente com os dados refinados desde o início, validando a eficácia do reaproveitamento de anotações menos precisas como uma etapa intermediária de aprendizado.

Por fim, é relevante ressaltar a dimensão reduzida dos conjuntos de dados utiliza-

dos: apenas 347 imagens para o *dataset* simples e 124 para o refinado. Esses resultados evidenciam dois pontos cruciais, primeiro, é possível obter bons resultados mesmo com datasets pequenos, e segundo, a qualidade das anotações tem maior impacto no desempenho final do que a mera quantidade de dados.

Corroborando a viabilidade de aplicação em *datasets* reduzidos, Zhang et al. [24] obtiveram êxito na extração de características fenotípicas de alfaces utilizando o Mask R-CNN. Embora os autores tenham adotado um *backbone* distinto, seus resultados alinham-se às conclusões de Osorio et al. [22] quanto à eficácia do modelo.

Para superar a limitação quantitativa das imagens originais, os autores empregaram técnicas de aumento de dados (*data augmentation*). Essa estratégia permitiu expandir artificialmente o conjunto de treinamento através de transformações nas imagens, garantindo maior variabilidade e robustez ao modelo.

Outro diferencial metodológico importante foi a adoção da validação cruzada (*k-fold cross-validation*) dividida em 5 partes ($k = 5$). Essa estratégia permitiu maximizar o uso dos dados disponíveis e mitigar possíveis vieses de seleção, evitando que a avaliação do modelo ficasse restrita a um conjunto fixo de treinamento e teste.

3 MÉTODO DE PESQUISA

3.1 Aquisição e Tratamento de Dados

A aquisição das imagens foi realizada através de fotografias digitais com resolução original de 3000×4000 pixels. Posteriormente, estas imagens foram submetidas a um pré-processamento, sendo redimensionadas para a resolução de 512×683 pixels e padronizadas no formato PNG.

As coletas foram conduzidas em ambiente real de comercialização (mercados), sob condições de iluminação artificial variada. Um fator relevante deste conjunto de dados é a complexidade do cenário, que apresenta um fundo variável composto por elementos ruidosos como chão, prateleiras e outras frutas adjacentes, simulando as condições reais de aplicação do modelo.

O *dataset* final é constituído por 300 imagens, que variam desde maçãs com avarias severas e visíveis até frutos aparentemente sadios. Além disso, as amostras apresentam diversidade na coloração da epiderme, incluindo tons de verde, amarelo e vermelho. Para garantir a robustez estatística dos experimentos, a partição dos dados não seguiu uma divisão estática; em vez disso, adotou-se a técnica de validação cruzada (*k-fold cross-validation*). Essa abordagem permite utilizar a totalidade dos dados para treinamento e validação em diferentes iterações, mitigando vieses de seleção.

Para a definição das categorias de detecção, o modelo foi configurado para um problema de classe única. Desta forma, definiram-se apenas duas classes, a classe 0, representando o fundo da imagem (*background*), e a classe 1, correspondente ao objeto de interesse, ou seja, os danos físicos presentes na epiderme da maçã.

A geração das máscaras de segmentação (*ground truth*) foi realizada manualmente, utilizando uma ferramenta de anotação desenvolvida especificamente para esta pesquisa. O *software* permite a demarcação precisa através de polígonos, ajustando-se aos contornos irregulares das avarias. Para garantir a compatibilidade com o treinamento do modelo, a ferramenta foi projetada para exportar as anotações seguindo rigorosamente o padrão de dados do *VGG Image Annotator* (VIA), gerando arquivos no formato JSON contendo as coordenadas espaciais de cada região.

Para garantir a diversidade das amostras e simular as técnicas de aumento de dados (*data augmentation*), as operações de rotação e inversão foram realizadas manualmente no momento da captura. Esta estratégia envolveu a variação física da orientação da fruta e da câmera em cada registro, assegurando que o modelo fosse exposto a diferentes perspectivas da maçã e dos danos, aumentando a capacidade de generalização da rede.

Também foi aplicada essa estratégia de forma artificial, gerando um novo *dataset* com tamanho 5× maior. Esse *dataset* foi utilizado em um experimento específico a fim de validar a melhoria dos resultados utilizando essa técnica.

3.2 Configuração do modelo de treinamento

O treinamento do modelo de rede neural foi baseado na arquitetura Mask R-CNN, que é reconhecida por sua segmentação de instâncias.

O modelo Mask R-CNN, utilizando o ResNet-101 como *backbone*, foi inicializado com pesos pré-treinados no grande *dataset* COCO [23]. Esta abordagem de *transfer learning* é crucial para otimizar o processo de convergência, reduzindo o tempo de treinamento necessário e auxiliando o modelo a alcançar alto desempenho com um *dataset* de dimensão reduzida (300 imagens).

Em função dessa estratégia de reaproveitamento de conhecimento, a taxa de aprendizado (*Learning Rate* - *LR*) foi ajustada dinamicamente. Utilizou-se um *LR* mais elevado de 0.001 nas etapas iniciais (ajuste apenas das camadas de predição), enquanto um *LR* significativamente menor, de 0.0001, foi aplicado nos ciclos de *fine-tuning* que envolveram o treinamento de todas as camadas. Essa redução final visa preservar o conhecimento robusto já capturado pelo ResNet-101, assegurando que as modificações nos pesos sejam feitas de forma cautelosa e incremental.

O processo de otimização foi conduzido utilizando o algoritmo *Stochastic Gradient Descent* (SGD). Esta escolha é amplamente adotada em arquiteturas R-CNN devido à sua eficácia na navegação pelo espaço de perdas e sua comprovada robustez.

Foi configurado um *Momentum* de **0.9** (*LEARNING_MOMENTUM*), que é crucial para acelerar a convergência em direções relevantes e amortecer as oscilações do gradiente. Decaimento de Peso (*Weight Decay*): Aplicou-se um valor de **0.0001**. Este parâmetro atua como um termo de regularização *L2*, que penaliza pesos muito grandes, mitigando o risco de sobreajuste (*overfitting*) nos dados.

O tamanho do lote (*batch size*) foi definido como uma (1) imagem por iteração. O número total de iterações por época foi estabelecido em 240, o que corresponde a 80% do dataset total, assegurando que o conjunto de treinamento fosse utilizado integralmente em cada ciclo. Paralelamente, foram definidas 60 etapas de validação, garantindo que todas as imagens do conjunto de validação fossem avaliadas ao final de cada época. Por fim, o treinamento foi definido para 100 épocas, a fim de monitorar e avaliar detalhadamente o comportamento da função de perda e a convergência do modelo.

O modelo final é treinado para minimizar o valor agregado (L), conforme a Equação

3.1:

$$L = L_{RPN_{cls}} + L_{RPN_{box}} + L_{MRCNN_{cls}} + L_{MRCNN_{box}} + L_{MRCNN_{mask}} \tag{3.1}$$

Conforme a configuração adotada, o peso de contribuição para cada um dos cinco componentes de perda foi ajustado para 1.0, garantindo que a rede atribua igual importância a todas as tarefas$L_{RPN_{cls}}$ e $L_{RPN_{box}}$ perdas pela classificação e ajuste da caixa das propostas de região.$L_{MRCNN_{cls}}$ e $L_{MRCNN_{box}}$ perdas pela classificação final do objeto e pelo refinamento das coordenadas da caixa delimitadora.$L_{MRCNN_{mask}}$ perda pela segmentação do objeto, calculada separadamente para cada classe (para evitar competição entre as classes na predição da máscara).

3.3 Métricas

A avaliação do desempenho de modelos de segmentação de instâncias exige métricas robustas que combinem precisão na classificação e acurácia na localização espacial. Para este trabalho, as métricas escolhidas foram selecionadas para refletir a capacidade do modelo de identificar o dano e segmentar seu contorno de forma precisa.

As métricas utilizadas para a análise de desempenho incluem o *Intersection Over Union* (IoU), o F1-*Score* e, como métrica agregada principal, o *Mean Average Precision* (mAP). A Função de Perda total do Mask R-CNN será monitorada durante o treinamento como uma ferramenta de diagnóstico para avaliar a convergência e o ajuste dos pesos.

3.3.1 Intersection Over Union

Essa métrica demonstra o nível de precisão da localização do objeto detectado em relação à área de referência do conjunto de validação. Como podemos ver na Figura 10 O IoU é a razão entre a Área de Intersecção, (a de sobreposição das áreas em azul escuro), e a Área de União (o espaço total abrangido pelas máscaras real e predita). A área delimitada pela borda verde representa a máscara real ($\mathbf{M_{gt}}$), e a área delimitada pela borda vermelha representa a máscara predita ($\mathbf{M_p}$).

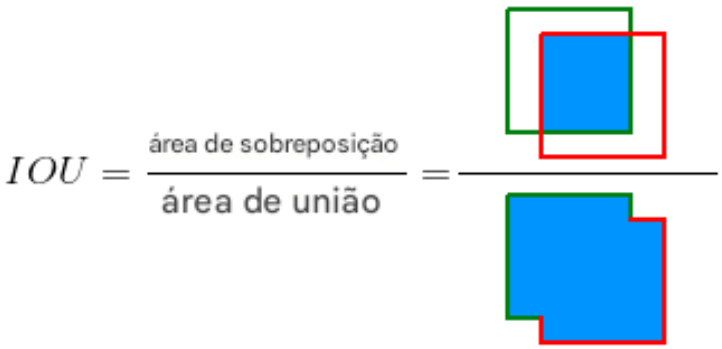

Figura 10 – Representação do cálculo do *Intersection Over Union* (IoU).

A *Intersection Over Union* (IOU) é uma métrica fundamental utilizada no contexto de detecção de objetos para avaliar a similaridade e a qualidade da correspondência entre a área detectada (*bounding box* prevista, BP) e a área real do objeto alvo *ground truth* (*bounding box* real, BR). Essencialmente, a IOU verifica quão bem a localização e o tamanho da caixa prevista se alinham com a caixa real.

O cálculo da métrica IoU para Máscaras em vez de *bounding boxes* consiste em identificar primeiramente a área de sobreposição, ou intersecção, entre as duas máscaras, sendo elas a predita (M_P) e a real (M_R). Na sequência, determina-se a área total ocupada pelas duas máscaras juntas, denominada união. A métrica é então obtida através da divisão da área da intersecção pela área da união, resultando em um valor adimensional que varia entre 0 e 1.

Um resultado igual a 1 indica uma correspondência perfeita, enquanto o valor 0 denota que as máscaras não se interceptam em nenhum ponto. Quanto mais próximo o coeficiente estiver de 1, superior é considerada a qualidade da segmentação. É importante notar que essa comparação só possui validade estatística quando realizada entre máscaras reais e predições pertencentes à mesma classe.

A aplicação do IoU é fundamental para classificar uma detecção como correta baseando-se em um limiar de corte, ou *threshold*. Um limiar próximo de 1, como 0.75, impõe um critério restritivo que exige sobreposição quase total para validar o acerto. Em contrapartida, limiares mais próximos de 0 oferecem maior flexibilidade, aceitando detecções com sobreposições parciais entre a predição e o valor real [25]. Por fim, para uma análise global do desempenho do modelo, utiliza-se a média do IoU de todas as imagens (mIoU), avaliada em conjunto com a confiabilidade da região delimitada, a qual é mensurada submetendo as predições a variadas faixas de limiares.

3.3.2 Average Precision

O cálculo das métricas de desempenho baseia-se inicialmente na qualidade da localização, mensurada pelo *Intersection over Union* (IoU).

Para definir se uma detecção é válida, aplica-se um limiar (*threshold*, geralmente $\alpha = 0.5$). Se o IoU calculado for superior a este limiar, a detecção é considerada correta. Em cenários onde o modelo gera múltiplas predições para um mesmo objeto, considera-se aquela com a maior intersecção válida, conforme ilustrado na Figura 11, onde é composta por três imagens: a imange mais à esquerda mostra a predição do modelo de duas áreas em ciano e vermelho, a mais à direita mostra a máscara real em vermelho, e no meio, a interseção entre a predição que cobriu a maior área da máscara real, destacada em verde.

Figura 11 – Exemplo de seleção da melhor interseção de IoU para validação da detecção.

Com base nesse critério de IoU, as predições são classificadas em três categorias:

- Verdadeiro Positivo (VP): O modelo detecta um dano existente com precisão de localização suficiente (IoU $\geq \alpha$).
- Falso Positivo (FP): O modelo prevê um dano onde não existe (alarme falso) ou a localização é imprecisa (IoU $< \alpha$).
- Falso Negativo (FN): Existe um dano real na imagem, mas o modelo falha em detectá-lo.

Com base nessa classificação, obtêm-se duas métricas essenciais: a Precisão (Equação 3.2), que define o grau de certeza das detecções geradas pelo modelo, e o *Recall* (Equação 3.3), que demonstra o quanto o sistema foi capaz de cobrir todas as instâncias reais presentes na imagem.

$$\text{Precisão} = \frac{VP}{VP + FP} \tag{3.2}$$

$$\text{Recall} = \frac{VP}{VP + FN} \tag{3.3}$$

Para sintetizar o desempenho do modelo, utiliza-se o *Average Precision* (AP). O cálculo desta métrica envolve ordenar todas as predições do conjunto de validação de forma decrescente pelo *score* de confiança. Com essa lista ordenada, calculam-se a Precisão e o *Recall* acumulados a cada nova predição, construindo a Curva Precisão-*Recall* (P-R).

A *Average Precision* (AP) corresponde à área sob esta curva. Essa área é calculada através de uma soma discreta, ponderando a precisão em cada incremento de *recall*. Matematicamente, o AP é definido pela equação 3.4.

$$\text{AP} = \sum_{k=1}^{n} (R_k - R_{k-1}) \times P_k \tag{3.4}$$

Onde

- n é o número total de predições.
- R_k e P_k são, respectivamente, o *Recall* e a Precisão da predição k k.

.

3.3.3 F1-Score

Enquanto a Precisão e o *Recall* fornecem visões isoladas sobre a confiabilidade e a sensibilidade do modelo, muitas vezes é necessária uma métrica única que sintetize o equilíbrio entre ambas. Para isso, utiliza-se o F1-*Score* (ou Medida-F).

Conforme definido por Sasaki et al. [26], o F1-*Score* é a média harmônica entre a precisão e o *Recall*. A escolha pela média harmônica, em vez de usar a média aritmética simples, deve-se à sua propriedade de penalizar valores extremos. Isso significa que, para o F1-*Score* ser alto, tanto a Precisão quanto o *Recall* precisam ser altos simultaneamente. Se uma das métricas for muito baixa (ex.: o modelo encontra todos os danos, mas gera muitos alarmes falsos), o F1-*Score* cairá drasticamente.

Matematicamente, a métrica é definida pela Equação 3.5:

$$\text{F1} = 2 \times \frac{\text{Precisão} \times \text{Recall}}{\text{Precisão} + \text{Recall}} \tag{3.5}$$

Onde:

- O valor resultante varia entre 0 e 1;
- Um valor próximo de 1 indica que o modelo possui excelente precisão e robustez na detecção (baixo índice de falsos positivos e falsos negativos).

O F1-*Score* é calculado para um limiar de IoU específico (geralmente 0.5), servindo como um indicador pontual da eficiência do modelo em equilibrar a detecção correta dos danos sem excesso de predições incorretas.

4 EXPERIMENTOS

4.1 Execução do Treinamento em Cenário com *Dataset* Original

Para avaliar o impacto da profundidade do ajuste fino (*fine-tuning*) e da taxa de aprendizado na detecção de danos, foram conduzidos dois experimentos independentes. Em ambos os cenários, a rede foi inicializada com os pesos pré-treinados do *dataset* MS COCO [23], garantindo que os modelos partissem da mesma base de conhecimento genérico, sem que houvesse transferência de aprendizado sequencial entre as etapas experimentais.

Visando assegurar a robustez estatística dos resultados e mitigar possíveis vieses de seleção, a metodologia de validação cruzada (*k-fold cross-validation*) foi aplicada rigorosamente em ambos os experimentos. Para cada cenário, o conjunto de dados foi particionado em $k = 5$ subconjuntos distintos.

Consequentemente, o ciclo de treinamento foi realizado cinco vezes para cada configuração de modelo, alternando-se os conjuntos de treinamento e validação a cada iteração. Desta forma, os resultados finais de AP50 e AP75 apresentados neste trabalho não refletem uma execução isolada, mas sim a média aritmética obtida através das cinco validações, oferecendo uma estimativa confiável e imparcial da capacidade de generalização da rede.

4.1.1 Primeiro cenário experimental

Denominado Treinamento das Camadas de Predição, o processo restringiu-se exclusivamente às camadas superiores da rede, conhecidas como *heads*. Estas camadas são responsáveis pela proposta de regiões, classificação e geração das máscaras finais. Nesta configuração, os pesos do *backbone* ResNet-101 foram mantidos congelados (*frozen*), impedindo a atualização dos parâmetros das camadas extratoras de características profundas. Para esta etapa, definiu-se uma taxa de aprendizado (*Learning Rate*) de 0.001.

O valor mais elevado justifica-se pela necessidade de ajustar rapidamente os pesos das novas camadas, que são inicializados aleatoriamente, sem o risco de degradar as características extraídas pelo *backbone* estático. O objetivo principal deste experimento foi verificar se as características genéricas extraídas pela ResNet-101 seriam suficientes para descrever os danos nas maçãs, ajustando apenas a camada final de decisão.

4.1.2 Segundo cenário experimental

Neste cenário, realizou-se o treinamento completo da Rede, consistindo na atualização de toda a arquitetura. Diferentemente do anterior, todas as camadas foram des-

congeladas (*unfrozen*) desde o início do processo, permitindo a propagação do gradiente e a atualização dos pesos em toda a profundidade da rede.

Para este experimento, a taxa de aprendizado foi reduzida para 0.0001. Essa redução constitui uma medida de segurança essencial no *fine-tuning* profundo, visando preservar o conhecimento prévio do *backbone* enquanto se realizam ajustes finos e cautelosos nas características semânticas. O objetivo central foi permitir que a rede aprendesse padrões específicos da textura e das bordas dos danos, que são muito diferentes dos objetos comuns encontrados no *dataset* COCO, integrando esse conhecimento desde as camadas base até a saída.

4.2 Execução do Treinamento em Cenário com *Dataset* Expandido

Nesta etapa experimental, o foco deslocou-se para a avaliação do impacto do volume de dados no desempenho do modelo. Para isso, utilizou-se o conjunto de dados expandido artificialmente, gerado através das técnicas de aumento de dados (*data augmentation*) detalhadas na metodologia, tais como rotação e espelhamento. Assim como nos cenários anteriores, a rede foi inicializada carregando os pesos pré-treinados do *dataset* MS COCO, garantindo uma base sólida de extração de características.

A estratégia de treinamento adotada para este cenário consistiu na liberação de todas as camadas da rede para atualização desde o início do processo. Contudo, diferentemente do ajuste fino realizado no *dataset* original, a taxa de aprendizado (*Learning Rate*) foi configurada em 0.001. Esta escolha visa explorar a capacidade de convergência da rede diante de uma maior diversidade de amostras, mantendo uma taxa de atualização de pesos mais agressiva para o treinamento da arquitetura completa.

Devido ao aumento substancial no número de imagens disponíveis, os parâmetros de duração da época foram ajustados proporcionalmente para assegurar que todo o conjunto de dados fosse processado a cada ciclo. Desta forma, definiram-se 1440 etapas de treinamento e 360 etapas de validação por época. Esse dimensionamento garante a cobertura integral das variações sintéticas e originais, permitindo um monitoramento preciso da função de perda e das métricas de avaliação ao longo do treinamento.

Vale ressaltar uma alteração metodológica específica para este cenário. Diferentemente dos experimentos anteriores, não foi aplicada a técnica de validação cruzada (*k-fold cross-validation*). Esta decisão justifica-se pelo aumento substancial no volume de dados e, consequentemente, na carga computacional exigida.

Considerando que o *dataset* expandido aumentou o número de etapas por época para 1440, a execução de cinco ciclos completos de treinamento (como exigido pelo $k = 5$) tornaria o tempo de processamento inviável dentro do escopo deste trabalho. Por isso,

optou-se por uma divisão fixa entre treino e validação (80% para treinamento e 20% para validação).

5 RESULTADOS

Neste capítulo, são apresentados e discutidos os resultados experimentais obtidos na detecção e segmentação de danos em maçãs. A análise está estruturada de acordo com os cenários definidos na metodologia, primeiramente, avalia-se o desempenho no cenário com *dataset* original, comparando as estratégias de treinamento das camadas de topo (*head*) versus o modelo (*all*), com validação cruzada (*k-fold*).

Posteriormente, analisam-se os impactos do aumento de dados (*data augmentation*) no cenário com *dataset* expandido. Por fim, realiza-se uma análise qualitativa visual das máscaras geradas para validar a aplicabilidade do modelo.

Após o ciclo de treinamento, obtiveram-se os resultados da função de perda para os conjuntos de treino e validação. Inicialmente, analisa-se o resultado do experimento com *k-fold* mantendo o congelamento (*freezing*) dos pesos, treinando apenas as camadas *heads*.

Pode-se observar na Figura 12 que tanto a perda de treino (*loss*) quanto a de validação (*val loss*) iniciaram com valores próximos a 1.5. Porém, com o avanço das épocas, as curvas distanciaram-se progressivamente. Embora a perda de validação tenha reduzido nas etapas iniciais, nota-se que, após a época 20, ela volta a crescer gradualmente, à primeira vista, indicando uma dificuldade de generalização do modelo.

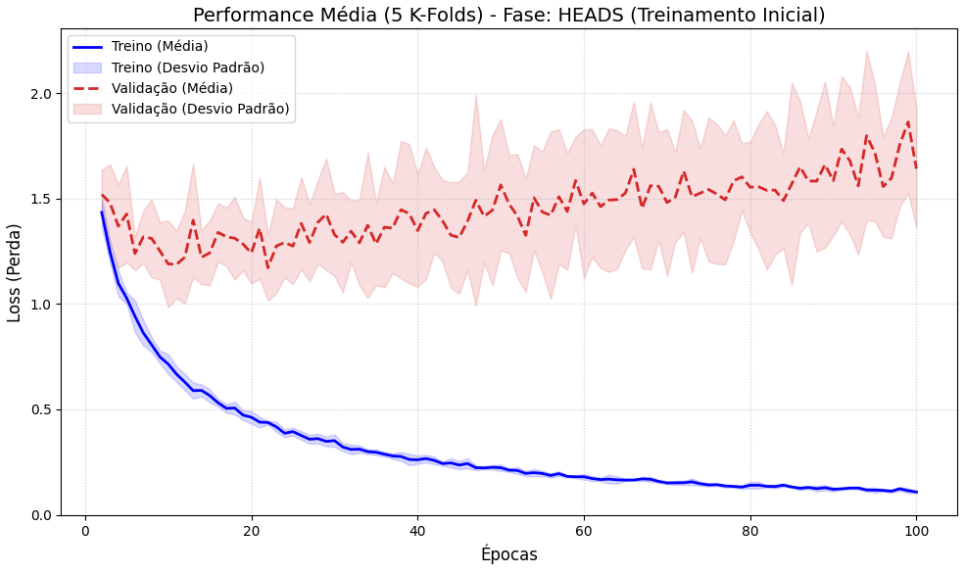

Figura 12 – Curvas de aprendizado da função de perda durante o treinamento do modelo *head*.

Ao decompor os componentes da função de perda, torna-se possível analisar detalhadamente o comportamento da convergência. A Figura 13 demonstra que, no conjunto

de treinamento, todas as perdas convergem gradualmente, conforme esperado.

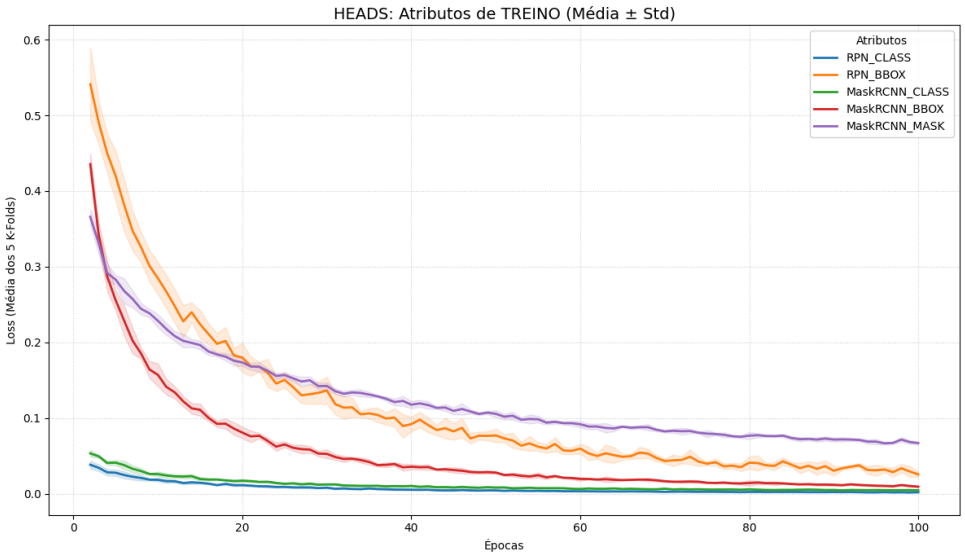

Figura 13 – Curvas de aprendizado dos elementos que compõem a perda total de treinamento do modelo *head*.

Entretanto, o cenário de validação apresentado na Figura 14, revela um comportamento distinto. Observa-se que as perdas de classificação e regressão (*bounding box*) convergiram levemente nas etapas iniciais e, em seguida, estagnaram. Em contrapartida, a perda da máscara convergiu levemente no início, mas logo passou a divergir gradualmente.

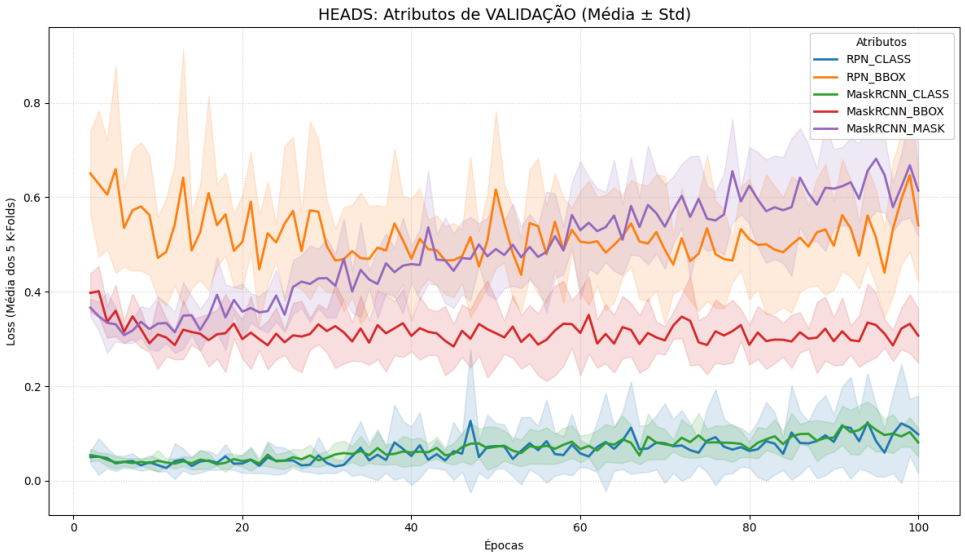

Figura 14 – Curva de aprendizado dos elementos que compõem a perda total de validação do modelo *head*.

Essa disparidade permite concluir que, embora o modelo mantenha uma boa precisão na localização da área do dano (validada pela estabilidade da *mrcnn bbox loss*),

ele perde progressivamente a capacidade de segmentação precisa. Ou seja, indicando que com o avanço do treinamento, a rede falha em segmentar com precisão a máscara real de validação. Porém devemos levar em consideração que a proporção de tamanho dos danos referentes a imagem são relativamente pequenos, fazendo com que qualquer variação mínima entre a mascara real e a predição do modelo elevem o erro da mascara.

Prosseguindo para a análise do treinamento completo do modelo (*All*), observa-se na Figura 15 um padrão comportamental similar ao cenário anterior, caracterizado pela divergência entre as curvas de treino e validação. Contudo, uma diferença crucial reside na magnitude dos valores. O treinamento começou com perdas abaixo de 1.0, o que sugere um ajuste rápido do modelo ao conjunto de dados do que o modelo anterior.

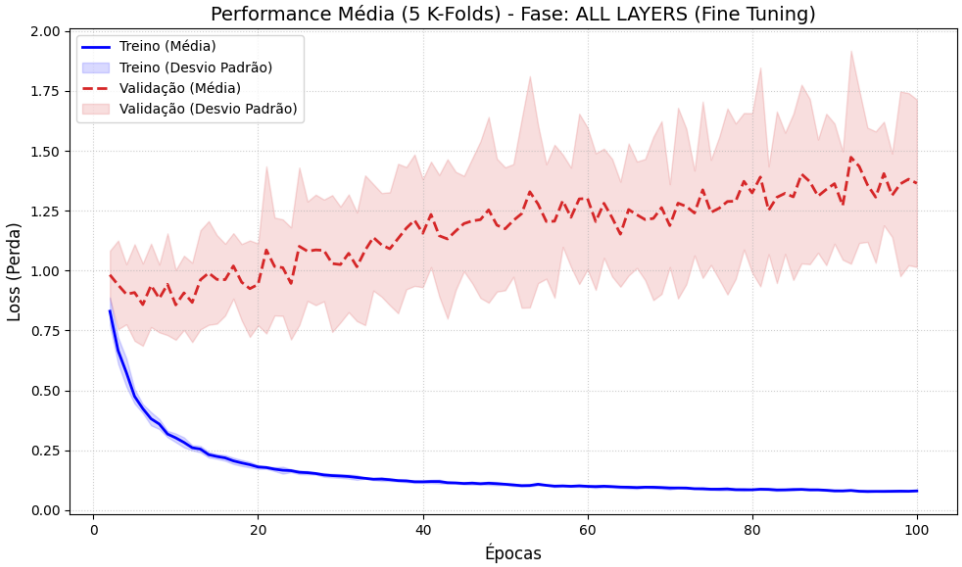

Figura 15 – Curvas de aprendizado da função de perda durante o treinamento do modelo *all*.

Embora a perda de validação (*val loss*) do modelo *All* tenha apresentado crescimento gradual conforme o avanço das épocas, seus valores absolutos mantiveram-se majoritariamente abaixo de 1.5, estado que correspondia apenas ao início do treinamento no cenário do modelo *Head*. Isso sugere que o descongelamento das camadas profundas permitiu refinar a extração de características, resultando em um erro final menor.

Decompondo o *loss* e *val loss*, podemos rever o mesmo cenário que do modelo *Head*, a perda no treinamento convergiu gradualmente sem problemas, porém com valores iniciais menores que no treinamento do modelo *Head*. O mesmo comportamento ocorre com a perda de validação, apresentando valores absolutos mais baixos, mas mantendo a tendência de crescimento da perda da mascara, como podemos ver nas Figuras 16 e 17.

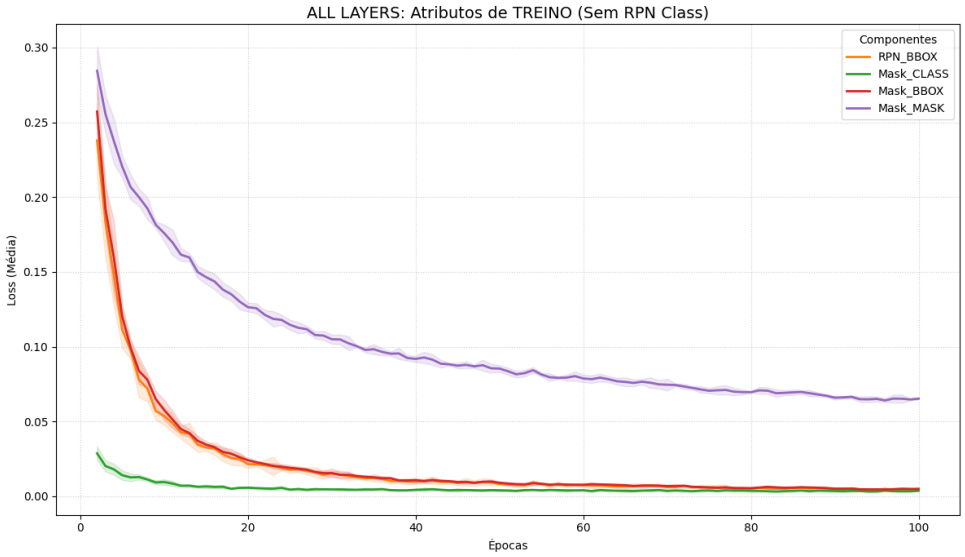

Figura 16 – Curvas de aprendizado dos elementos que compõem a perda total de treino do modelo *all* (sem o *RPN Class loss*).

Nota: O componente *RPN Class loss* foi suprimido do gráfico pois apresentou valores discrepantes (*outliers*) em algumas épocas, o que distorcia a escala e impossibilitava a visualização correta dos demais elementos.

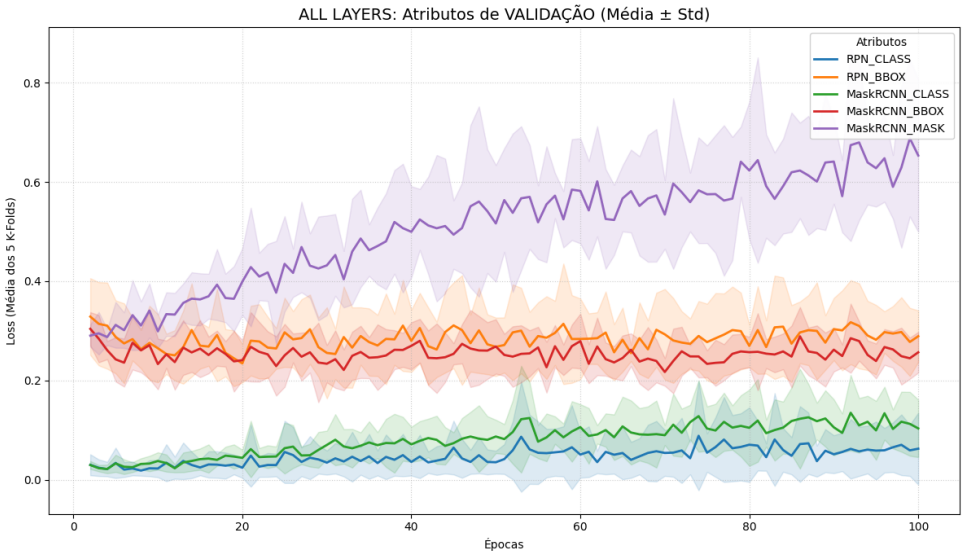

Figura 17 – Curvas de aprendizado dos elementos que compõem a perda total de validação do modelo *all*.

A análise das curvas de perda do modelo treinado com o *dataset* aumentado, denominado de*Expanded*, revela uma evolução substancial. A perda de validação estabilizou-se em patamares inferiores a 0.8 como demonstrado na Figura 18, contrastando com os cenários anteriores, onde o modelo *Head* atingiu valores próximos a 1.5 e o modelo *All* aproximou-se de 1.25. Essa redução drástica na perda de validação representa um ganho de desempenho de $53,3\%$ quando comparado ao cenário base (*Head*), evidenciando a eficácia do aumento de dados na adaptabilidade do modelo.

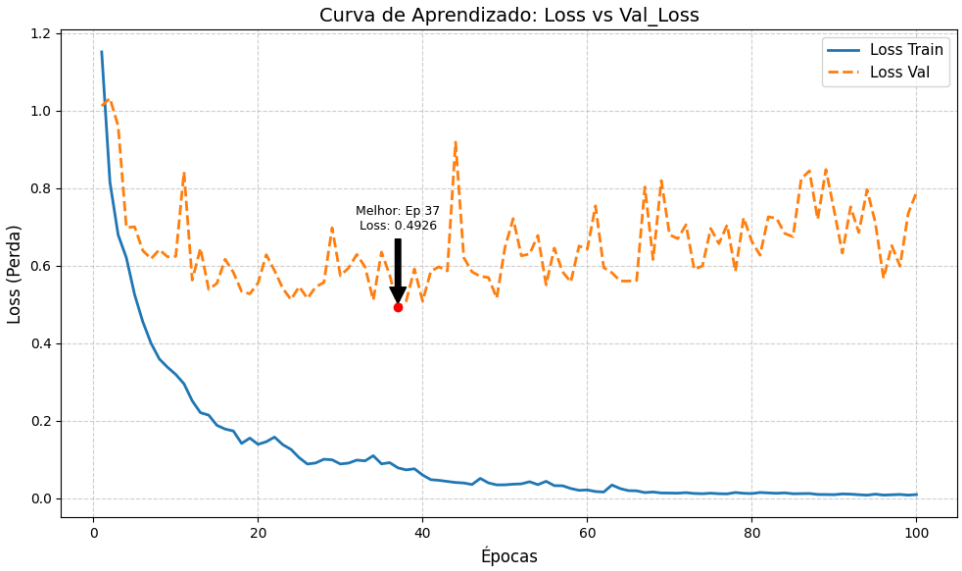

Figura 18 – Curvas de aprendizado da função de perda durante o treinamento do modelo *expanded*.

A figura destaca a época 37 como o ponto de mínima validação, onde a perda de treinamento atingiu 0.4926.

A decomposição do erro de validação, apresentada na Figura 19, revela que o componente de maior impacto foi a perda de regressão das propostas de região (*RPN Box Loss*). As perdas de classificação estagnaram, apresentando uma leve divergência nas épocas finais, mas mantiveram-se majoritariamente abaixo de 0.1. O destaque principal, contudo, reside nas perdas de *Bounding Box* e da Máscara, que inicialmente convergiram gradualmente e estabilizaram.

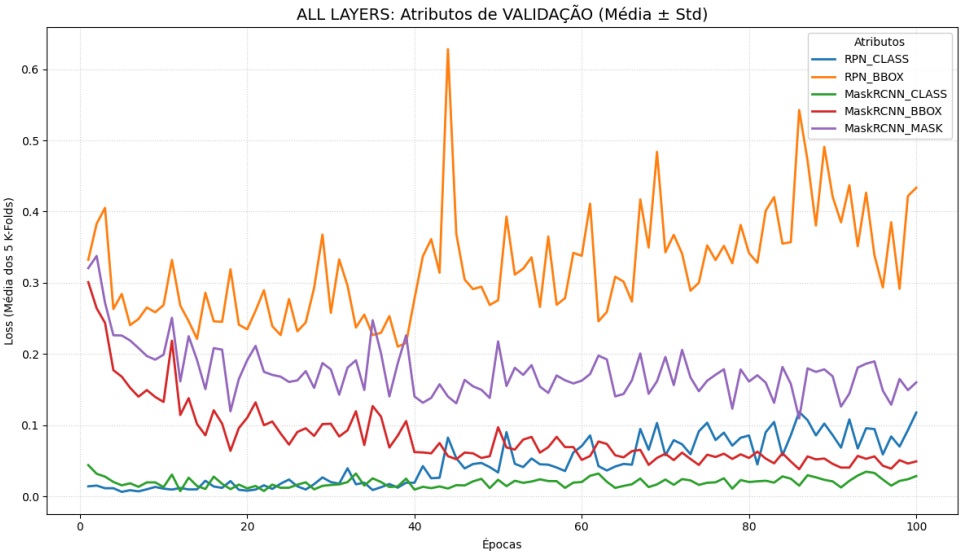

Figura 19 – Curvas de aprendizado dos elementos que compõem a perda total de validação do modelo *expanded*.

Ao analisar as curvas de perda de treinamento na Figura 20, observou-se uma divergência sequencial no treinamento: a partir da época 40, a perda de classificação (*MaskRCNN CLASS*) desestabilizou-se, seguida pelo colapso da RPN (*RPN CLASS*) após a época 60 e da regressão das caixas (*MaskRCNN BBOX*) após a época 80. Essa instabilidade deve-se à taxa de aprendizado de 0.001, que se mostrou agressiva para a atualização simultânea de toda a rede. Portanto, a análise restringi-se às épocas estáveis anteriores a esses eventos.

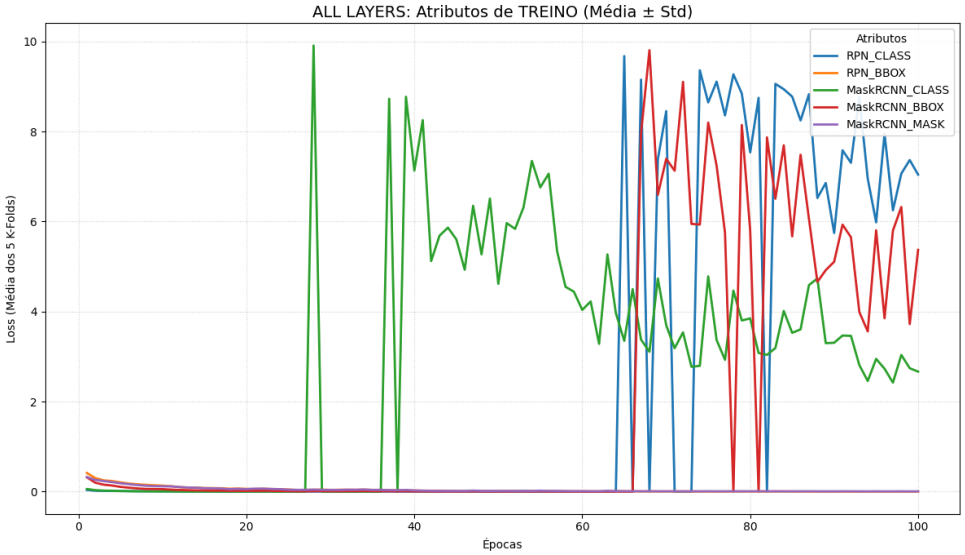

Figura 20 – Curvas de aprendizado dos elementos que compõem a perda total de treinamento do modelo *expanded*.

Avançando para a análise quantitativa, apresenta-se a comparação do desempenho dos modelos em relação às métricas estabelecidas. As Figuras 21, 22 e 23 apresentam os resultados obtidos em cada época para as métricas: *Average Precision* (AP@50) e *F1-Score@50* (ambas com *threshold* de 0.5), além da média de interseção sobre união (*mIoU*).

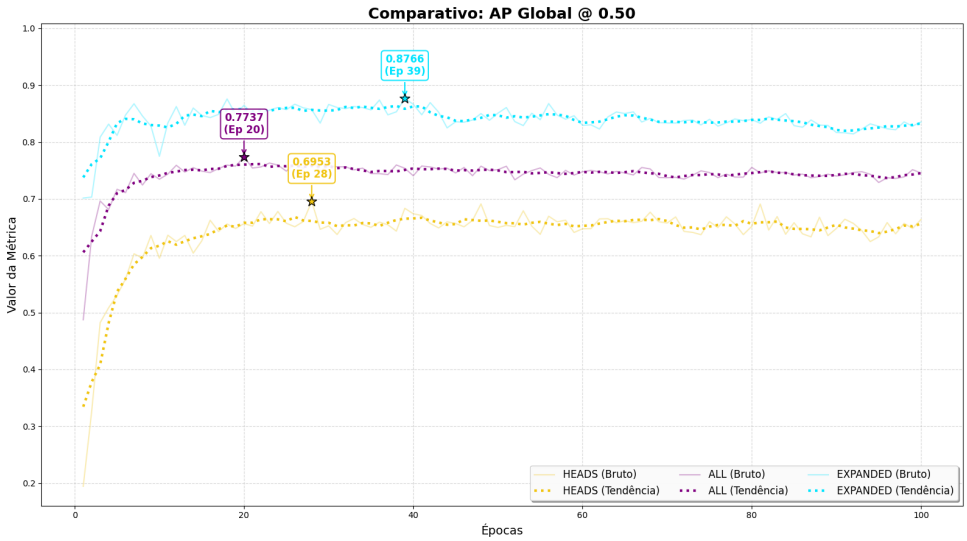

Figura 21 – Curvas de *Average Precision* (AP) calculadas com *threshold* de 0.5.

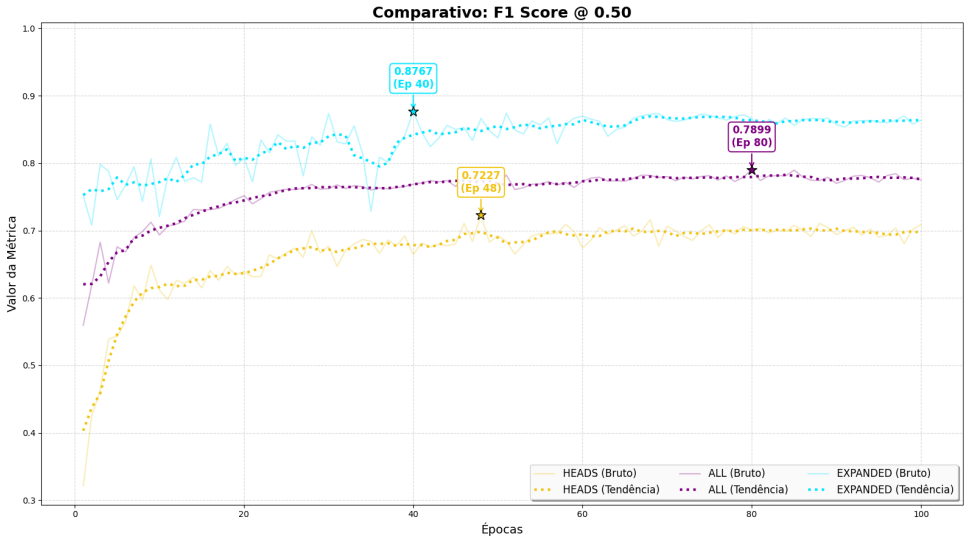

Figura 22 – Curvas de *F1 score* (F1) calculadas com *threshold* de 0.5.

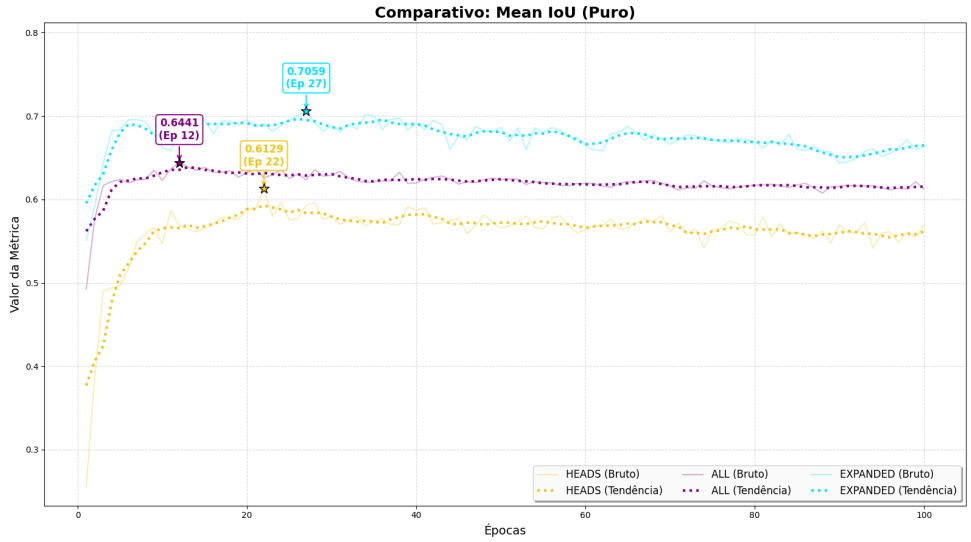

Figura 23 – Curvas de mIoU média de interseção sobre união

Conforme detalhado na Tabela 1, o modelo treinado com o *dataset* expandido adaptou-se melhor do que as abordagens anteriores. Enquanto o modelo *Head (Freezing)* estagnou com um mIoU de 0.6129 e o *All (Unfreezing)* atingiu 0.6441, o modelo *Expanded* alcançou a marca de 0.7059, demonstrando maior capacidade de segmentação. Vale ressaltar que, para o modelo *Expanded*, os dados reportados na tabela foram extraídos exclusivamente das épocas de estabilidade numérica, ignorando os picos de divergência causados por explosões de gradiente, a fim de garantir uma avaliação mais fiel do potencial da rede.

Tabela 1 – Comparativo detalhado dos melhores resultados obtidos por métrica e a respectiva época de ocorrência para cada modelo.

	Head (Freezing)		**All (Unfreezing)**		**Expanded**	
Métrica	**Valor**	**Ep.**	**Valor**	**Ep.**	**Valor**	**Ep.**
AP@50	0.6953	28	0.7737	20	**0.8757**	18
AP@75	0.2205	45	0.2681	38	**0.3139**	04
F1@50	0.7227	48	0.7899	80	**0.8735**	30
F1@75	0.3359	79	0.3882	91	**0.4269**	25
mIoU	0.6129	22	0.6441	12	**0.7059**	27

Nota: Valores em negrito indicam o melhor desempenho. "Ep."refere-se à época de treinamento onde o valor máximo foi registrado. Para o modelo *Expanded*, os valores foram extraídos apenas das épocas em que não houve explosões de gradiente.

Para a análise qualitativa das segmentações, as Figuras abaixo apresentam um comparativo visual entre as máscaras preditas e a anotação real (*ground truth*). A seleção dos pesos para a geração dessas inferências baseou-se estritamente no melhor desempenho de segmentação obtido.

Para os modelos treinados com validação cruzada, isolou-se o subconjunto (*fold*) de maior rendimento: no cenário *Head*, utilizou-se o *Fold* 4 na época 22 ($mIoU = 0.7051$); no cenário *All*, selecionou-se o *Fold* 4, com pico de desempenho na época 13 ($mIoU = 0.7255$). Já para o modelo *Expanded*, o melhor resultado estável foi registrado na época 27 ($mIoU = 0.7059$).

Na Figura 24 podemos observar o comparativo visual de segmentação. A imagem **original** apresenta o *ground truth* com duas máscaras reais destacadas em **ciano** e **vermelho**. As demais imagens ilustram a capacidade de detecção e o delineamento das máscaras geradas pelos modelos *Head*, *All* e *Expanded* para estas mesmas instâncias.

Figura 24 – Predição 1 dos modelos *Head*, *All* e *Expanded* em comparação com a mascara real.

Na Figura 25 observamos o comparativo de robustez a sombras. O modelo *Head* gerou um falso positivo (máscara ciano) ao confundir a sobra entre o dedo e a maçã com um defeito. Os modelos *All* e *Expanded* realizaram a detecção correta, ignorando sobra e segmentando apenas o dano real (vermelho).

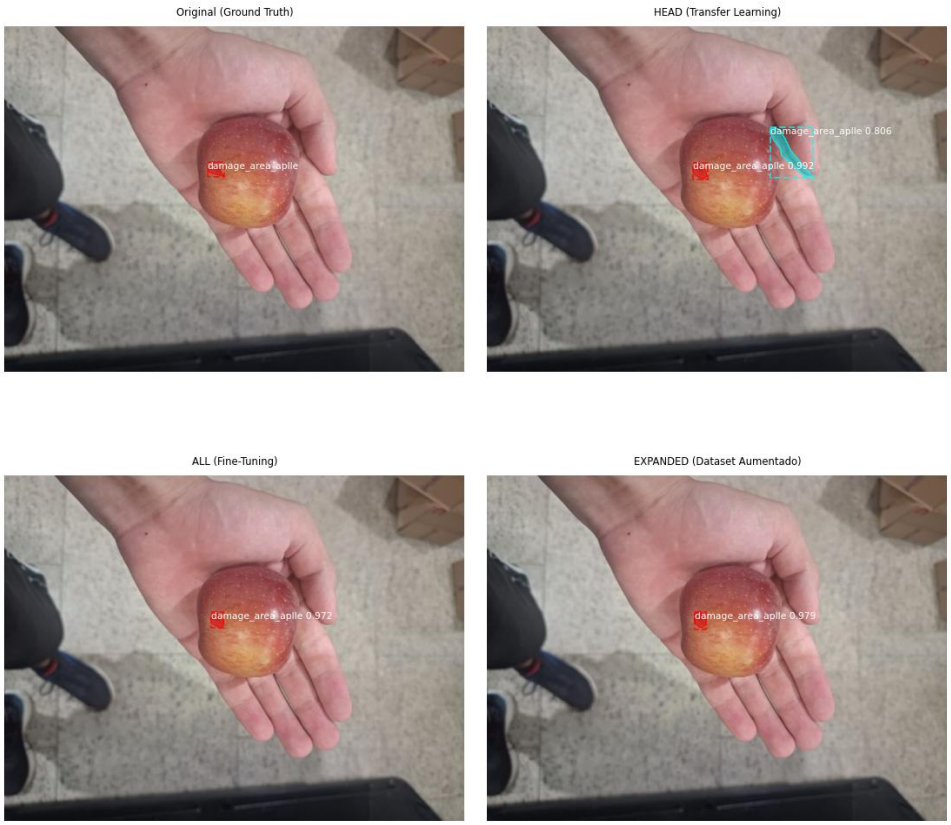

Figura 25 – Predição 2 dos modelos *Head*, *All* e *Expanded* em comparação com a mascara real.

E por fim na Figura 26 visualizamos o comparativo de detecção. Todos os modelos detectaram o defeito, porém o modelo *Expanded* apresentou uma capacidade de generalização superior, demonstrando maior precisão no delineamento da curvatura do dano em relação aos modelos *Head* e *All*.

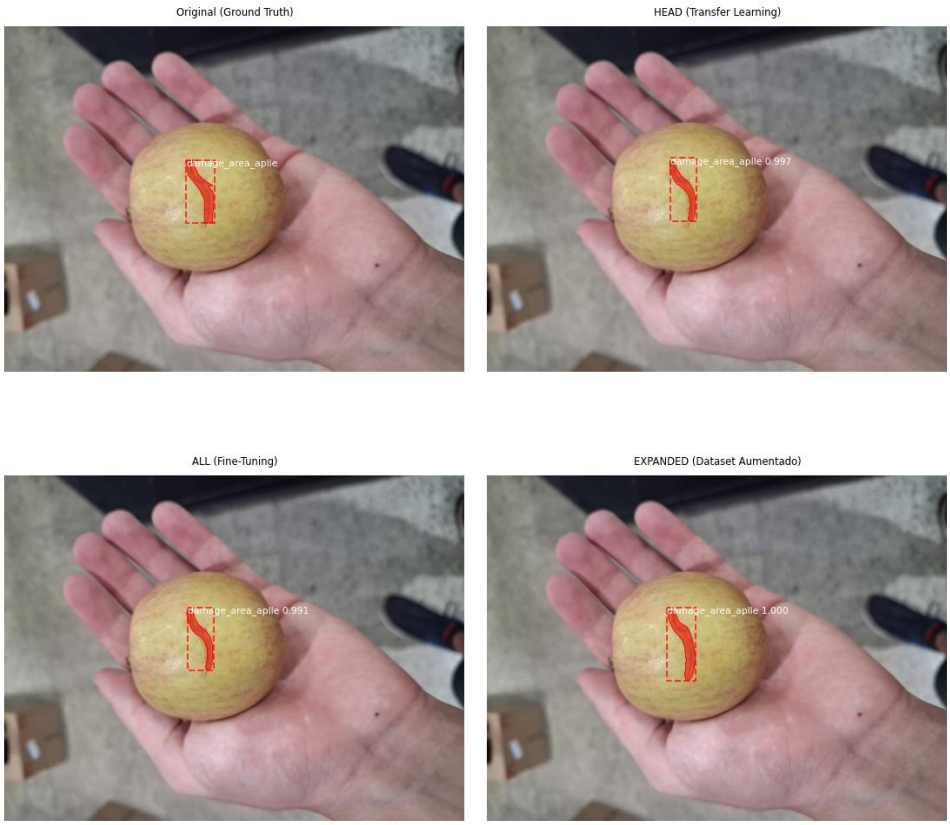

Figura 26 – Predição 3 dos modelos *Head*, *All* e *Expanded* em comparação com a mascara real.

6 CONCLUSÃO

O presente estudo cumpriu o objetivo de avaliar o desempenho e a utilidade da arquitetura Mask R-CNN na tarefa complexa de segmentação de instâncias de danos físicos em maçãs. A investigação comparativa permitiu identificar os limites da aplicação de Transfer Learning em datasets pequenos e a importância da variabilidade de dados para a precisão da segmentação.

A análise dos experimentos demonstrou que as estratégias baseadas exclusivamente no ajuste de pesos (*Head* e *All*) apresentaram limitações. Embora esses modelos tenham sido eficazes na localização espacial do defeito (detecção), a decomposição da função de perda de validação revelou uma dificuldade de delinear a forma precisa da avaria em realção as anotações espaciais reais, refletida na divergência persistente da perda de segmentação (L_{mask}). Identificou-se que essa dificuldade técnica decorre da relação de escala, como a proporção do dano físico é significativamente menor que a área total da imagem, pequenas imprecisões na predição da máscara geram penalizações desproporcionais nas métricas de validação.

Em contrapartida, a investigação confirmou que a capacidade de generalização do modelo está diretamente atrelada à exposição a variações espaciais durante o treinamento. A utilização do *dataset* expandido (Data Augmentation) provou ser a estratégia determinante para a estabilização do modelo. Esta abordagem permitiu que a rede aprendesse a variabilidade topológica dos defeitos, resultando em uma redução drástica no erro de segmentação e elevando o mIoU.

Conclui-se, portanto, que a arquitetura Mask R-CNN é uma ferramenta viável para auxiliar no cumprimento dos critérios relacionados a danos físicos da Instrução Normativa n.º 5/2006 [2]. O trabalho contribui para a modernização do setor frutícola ao demonstrar que, para além da simples detecção, é possível alcançar a segmentação automatizada necessária para mitigar a subjetividade humana no controle de qualidade de maçãs.

Para trabalhos futuros, propõe-se a otimização da arquitetura visando a execução do modelo em tempo real, permitindo sua integração em esteiras de classificação industrial. Recomenda-se também o desenvolvimento de métodos de pós-processamento para realizar o cálculo métrico da área dos danos segmentados (convertendo a contagem de pixels para mm^2 ou cm^2). Com base nessas medições, sugere-se a implementação de um algoritmo de decisão que aplique as tolerâncias da Instrução Normativa n.º 5/2006[2], classificando automaticamente as maçãs nas categorias Extra, Categoria I, Categoria II e Categoria III, consolidando assim uma solução completa para a automação do controle de qualidade.

REFERÊNCIAS

[1] IBGE. *Produção Agrícola Municipal: Culturas temporárias e permanentes: 2022*. Rio de Janeiro: Instituto Brasileiro de Geografia e Estatística, 2023. v. 49. 1–105 p. ISBN 978-65-87201-89-4.

[2] Brasil. Ministério da Agricultura, Pecuária e Abastecimento. *Instrução Normativa n.º 5, de 22 de fevereiro de 2006. Aprova o Regulamento Técnico de Identidade e Qualidade da Maçã*. 2006. Diário Oficial da União: seção 1, Brasília, DF, p. 3. Acesso em: 19 nov. 2025. Disponível em: <https://sistemasweb.agricultura.gov.br/sislegis/action/detalhaAto.do?method=visualizarAtoPortalMapa&chave=805793610>.

[3] CECHETTO, M. *Controle de qualidade durante a embalagem da maçã (Malus domestica) no Packing House da empresa Rasip*. 45 p. Dissertação (Trabalho de Conclusão de Curso (Agronomia)) — Universidade do Estado de Santa Catarina (UDESC), Lages, 2022. Disponível em: <https://repositorio.udesc.br/handle/UDESC/16800>.

[4] GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. Cambridge, MA: MIT Press, 2016. ISBN 9780262035613. Disponível em: <http://www.deeplearningbook.org>.

[5] BHARGAVA, A.; BANSAL, A. Fruits and vegetables quality evaluation using computer vision: A review. *Journal of King Saud University - Computer and Information Sciences*, Elsevier, v. 33, n. 6, p. 243–257, 2021.

[6] ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, v. 65, n. 6, p. 386–408, 1958.

[7] MINSKY, M.; PAPERT, S. A. *Perceptrons: An Introduction to Computational Geometry*. 1. ed. Cambridge, MA: The MIT Press, 1969. ISBN 0262130432.

[8] HAYKIN, S. *Neural Networks and Learning Machines*. 3. ed. Upper Saddle River, NJ: Pearson Education, 2009.

[9] SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Neural Networks*, Elsevier, v. 61, p. 85–117, 2015.

[10] YAMASHITA, R. et al. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, Springer, v. 9, p. 611–629, 2018.

[11] ESPINDOLA, A. C. *Computer vision and convolutional neural networks applied in pavement evaluation*. 220 f. p. Tese (Tese de Doutorado em Engenharia de Transportes) — Universidade Federal do Ceará (UFC), Fortaleza, 2024. Disponível em: <http://repositorio.ufc.br/handle/riufc/83018>.

[12] HE, K. et al. Deep residual learning for image recognition. In: IEEE. *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.], 2016. p. 770–778.

[13] GIRSHICK, R. et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2014. p. 580–587.

[14] UIJLINGS, J. R. R. et al. Selective search for object recognition. *International Journal of Computer Vision*, Springer, v. 104, n. 2, p. 154–171, 2013.

[15] CORTES, C.; VAPNIK, V. Support-vector networks. *Machine learning*, Springer, v. 20, n. 3, p. 273–297, 1995.

[16] GIRSHICK, R. Fast r-cnn. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2015. p. 1440–1448.

[17] REN, S. et al. Faster r-cnn: Towards real-time object detection with region proposal networks. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2015. v. 28.

[18] HE, K. et al. Mask r-cnn. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. [S.l.: s.n.], 2017. p. 2961–2969.

[19] LONG, J.; SHELHAMER, E.; DARRELL, T. Fully convolutional networks for semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 3431–3440.

[20] HOU, J. et al. Early bruise detection in apple based on an improved faster rcnn model. *Horticulturae*, MDPI, v. 10, n. 1, p. 100, 2024.

[21] AKROUCHI, M. E. et al. Optimizing mask r-cnn for enhanced quinoa panicle detection and segmentation in precision agriculture. *Frontiers in Plant Science*, Frontiers, v. 16, p. 1472688, 2025.

[22] OSORIO, K. et al. Mask R-CNN refitting strategy for plant counting and sizing in UAV imagery. *Remote Sensing*, MDPI, v. 12, n. 18, p. 3015, 2020.

[23] LIN, T.-Y. et al. Microsoft COCO: Common objects in context. In: SPRINGER. *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. [S.l.], 2014. p. 740–755.

[24] ZHANG, Y. et al. Study on utilizing mask R-CNN for phenotypic estimation of lettuce's growth status and optimal harvest timing. *Agronomy*, MDPI, v. 14, n. 6, p. 1271, 2024.

[25] PADILLA, R. et al. A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics*, MDPI, v. 10, n. 3, p. 279, 2021.

[26] SASAKI, Y. et al. The truth of the f-measure. In: *Teach Tutor Mater*. [S.l.: s.n.], 2007. v. 1, n. 5, p. 1–5.