Universidade de Brasília
Faculdade de Tecnologia

Classificação de imagens de insetos com poucas amostras: uma abordagem inovadora utilizando a arquitetura Swin

Gustavo Joshua de Faria Costa de Azevedo

PROJETO FINAL DE CURSO

ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Brasília

2023

Universidade de Brasília
Faculdade de Tecnologia

Classificação de imagens de insetos com poucas amostras: uma abordagem inovadora utilizando a arquitetura Swin

Gustavo Joshua de Faria Costa de Azevedo

Projeto Final de Curso submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação

Orientador: Prof. Dr. Díbio Leandro Borges

Brasília
2023

Faria Costa de Azevedo, Gustavo Joshua de.
F224c Classificação de imagens de insetos com poucas amostras: uma abordagem inovadora utilizando a arquitetura Swin / Gustavo Joshua de Faria Costa de Azevedo; orientador Díbio Leandro Borges. -- Brasília, 2023.
74 p.

Projeto Final de Curso (Engenharia de Controle e Automação) -- Universidade de Brasília, 2023.

1. Classificação de insetos. 2. Swin Transformer. 3. *Few-shot learning*. 4. Visão computacional. I. Borges, Díbio Leandro, orient. II. Título

Universidade de Brasília
Faculdade de Tecnologia

Classificação de imagens de insetos com poucas amostras: uma abordagem inovadora utilizando a arquitetura Swin

Gustavo Joshua de Faria Costa de Azevedo

Projeto Final de Curso submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação

Trabalho aprovado. Brasília, 08 de dezembro de 2023:

Prof. Dr. Díbio Leandro Borges, UnB/CIC
Orientador

Profa. Dra. Roberta Barbosa Oliveira, UnB/CIC
Examinador interno

Prof. Dr. Wilson Henrique Veneziano, UnB/CIC
Examinador interno

Brasília
2023

Agradecimentos

Gostaria de expressar minha profunda gratidão a todas as pessoas que contribuíram significativamente para a realização deste trabalho de graduação. Em especial, sou imensamente grato ao meu orientador prof. Díbio, cuja orientação foi fundamental para o desenvolvimento deste projeto. À minha amada namorada Graziela, que sempre esteve do meu lado com apoio emocional e compreensão, minha eterna gratidão. À minha família, que sempre acreditou em mim em todos os momentos, obrigado por tudo. Não posso deixar de agradecer a todos os amigos do curso de Engenharia Mecatrônica e da DROID que fiz durante a graduação, cuja colaboração foi essencial ao longo dessa longa jornada acadêmica.

Resumo

Este trabalho concentra-se na aplicação da arquitetura *Swin Transformer* na classificação de insetos danosos a plantações em situações em que há escassez de imagens. Com o advento da tecnologia e o crescimento da visão computacional, os métodos tradicionais de identificação de insetos passaram a ser substituídos por técnicas mais avançadas e precisas, com a utilização de inteligência artificial. A *Swin Transformer*, uma arquitetura inovadora que surge como uma alternativa às tradicionais redes neurais convolucionais e que se destaca na extração de características relevantes utilizando técnicas de janelas deslocadas, foi utilizada neste estudo para a classificação de insetos danosos a plantações em contextos agrícolas. A pesquisa avalia a eficácia de diferentes versões da *Swin Transformer*, explorando sua capacidade em cenários com um número limitado de imagens disponíveis, utilizando a metodologia *few-shot learning*. Através desta abordagem, o trabalho tem o objetivo de contribuir para o avanço do campo de classificação de insetos agrícolas, oferecendo uma avaliação abrangente das diferentes técnicas e seu potencial impacto na agricultura. Mesmo com quantidades pequenas de imagens, foram alcançados resultados de 96,05% de acurácia para classificação de insetos adultos, e 87,96% de acurácia para insetos em estágio inicial de vida utilizado testes *three-way five-shot*.

Palavras-chave: Classificação de insetos. Swin Transformer. *Few-shot learning*. Visão computacional.

Abstract

This work focuses on the application of the Swin Transformer architecture in the classification of insects pests in situations where there is a scarcity of images. With the advent of technology and the growth of computer vision, traditional methods of insect identification have been replaced by more advanced and accurate techniques involving artificial intelligence. The Swin Transformer, an innovative architecture that serves as an alternative to traditional convolutional neural networks, excels in extracting relevant features using shifted windows techniques. In this study, it was employed for the classification of harmful insects in agricultural contexts. The research evaluates the effectiveness of different versions of the Swin Transformer, exploring its capability in scenarios with a limited number of available images, utilizing the few-shot learning methodology. Through this approach, the work aims to contribute to the advancement of the agricultural insect classification field by providing a comprehensive assessment of different techniques and their potential impact on agriculture. Even with small quantities of images, results of 96.05% of accuracy were achieved for the classification of adult insects, and 87.96% of accuracy for insects in the early life stage, using three-way five-shot tests.

Keywords: Pest insects classification. Swin Transformer. Few-shot learning. Computer vision.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

Sumário

1 Introdução

1.1 Contextualização

A agricultura é uma das atividades mais antigas e fundamentais da humanidade. Ao longo dos anos, a agricultura tem evoluído significativamente, impulsionada por avanços tecnológicos que têm revolucionado como os agricultores cultivam e gerenciam suas terras. Um desses avanços notáveis é a agricultura de precisão, que consiste em um conjunto de técnicas que permitem o gerenciamento localizado dos cultivos por meio de tecnologias avançadas (Anghelof *et al.*, 2020).

A agricultura de precisão experimenta notáveis avanços com o monitoramento inteligente de insetos danosos a plantações, uma técnica essencial que substitui os métodos tradicionais de detecção manual. Anteriormente, a identificação de insetos demandava muito esforço humano e especialização para distinguir entre diferentes espécies, tarefa complicada devido à semelhança entre algumas delas (Høye *et al.*, 2021). No entanto, a introdução do monitoramento inteligente, que incorpora tecnologias avançadas como sensores, câmeras e algoritmos de inteligência artificial, revoluciona esse cenário.

Essas tecnologias automatizadas, impulsionadas por inteligência artificial, conseguem identificar e rastrear insetos com precisão, eliminando a necessidade de abordagens manuais (Li *et al.*, 2021a). Um benefício adicional é a capacidade de lidar com espécies de insetos muito semelhantes, tornando o reconhecimento manual uma prática cada vez mais rara. Ao empregar o monitoramento inteligente, os agricultores têm a capacidade de detectar rapidamente infestações e tomar medidas imediatas, minimizando assim os danos às plantações e reduzindo a dependência do uso indiscriminado de pesticidas.

Essa abordagem eficiente e sustentável não apenas preserva o meio ambiente, mas também contribui para o aumento da produtividade e rentabilidade nas atividades agrícolas. Ao promover a rápida resposta contra infestações, os agricultores conseguem proteger suas plantações de maneira mais eficaz, criando um cenário onde a necessidade de pesticidas é reduzida. Assim, o monitoramento inteligente, aliado à inteligência artificial, não apenas aprimora a gestão de pragas de insetos, mas também promove práticas agrícolas mais responsáveis, alinhadas com a preservação ambiental e o aumento da eficiência no setor agrícola.

Dentre esses algoritmos de inteligência artificial, podem ser citados os algoritmos de aprendizado de máquina que têm recentemente despertado grande interesse devido à sua habilidade em lidar com grandes volumes de dados. No entanto, é importante destacar que esses algoritmos são altamente dependentes de recursos criados manualmente, o que implica

na necessidade de conhecimento humano e parâmetros complexos durante o processo de desenvolvimento. Quando utilizados em ambientes externos não controlados, esses métodos frequentemente enfrentam desafios na extração de recursos eficazes, o que pode resultar em modelos com ajuste inadequado, pouca robustez e capacidade de generalização limitada (Li *et al.*, 2021a).

Devido a essas limitações dos algoritmos tradicionais de aprendizado de máquina, os algoritmos de aprendizado profundo, especialmente as redes neurais convolucionais, ganharam ampla popularidade e são amplamente utilizados em implementações de visão computacional, como em Liu e Wang (2020), Kasinathan, Singaraju e Uyyala (2021) e Thenmozhi e Srinivasulu Reddy (2019). Essas redes têm se destacado por sua capacidade de extrair características relevantes e realizar tarefas de classificação, detecção e segmentação de objetos de maneira eficaz.

No entanto, uma evolução significativa ocorreu com o trabalho da *Swin Transformer* (Liu *et al.*, 2021). Tradicionalmente utilizados em tarefas de processamento de linguagem natural, os *transformers* foram aplicados à visão computacional e uma dessas abordagens é a inovadora arquitetura *Swin*. Essa arquitetura inovadora baseada em *transformers* tem ganhado destaque recentemente. Ao utilizar janelas deslocadas para capturar informações espaciais, a *Swin Transformer* tem alcançado bons resultados em tarefas de visão computacional, como detecção de objetos e classificação de imagens, como pode ser visto no artigo base (Liu *et al.*, 2021). Sua aplicação tem sido amplamente explorada e avaliada nos últimos anos, demonstrando um grande potencial para aprimorar o desempenho e expandir as possibilidades da visão computacional.

1.2 Descrição do problema e objetivos

Os insetos danosos são um dos principais desafios enfrentados pelos produtores agrícolas, pois podem causar danos consideráveis às plantações, resultando em perdas significativas na produção. A classificação precisa desses insetos é fundamental para implementar medidas de controle efetivas. No entanto, a identificação manual dos insetos é um processo demorado e propenso a erros, especialmente em grandes áreas de plantio.

Com os avanços tecnológicos, há uma demanda crescente por soluções escaláveis e automatizadas para a classificação de insetos em plantações. Nesse contexto, técnicas de visão computacional e aprendizado de máquina têm se mostrado promissoras. Essas técnicas podem acelerar e aprimorar a identificação de insetos, permitindo uma resposta mais rápida e precisa por parte dos produtores.

Portanto, este trabalho visa investigar a eficácia da *Swin Transformer* (Liu *et al.*, 2021) como uma arquitetura de rede neural para a classificação de insetos danosos a plantações em imagens agrícolas. A *Swin Transformer* é uma arquitetura recente que se destaca como

uma alternativa às redes neurais convolucionais tradicionais.

Com o propósito de avaliar a eficácia da *Swin Transformer* na classificação de insetos em imagens agrícolas, este estudo visa apresentar uma análise detalhada do desempenho dessa arquitetura. A *Swin Transformer* representa uma abordagem recente no campo do reconhecimento de padrões visuais, sendo inovadora, pois surge como uma alternativa ao uso predominante de redes neurais convolucionais, que têm sido amplamente dominantes nos últimos anos para essa tarefa.

Dada a complexidade e relevância do desafio associado à classificação de insetos em plantações, a pesquisa deste trabalho explora a aplicação da *Swin Transformer* como extratora de características. Diante da dificuldade em obter um conjunto robusto de imagens de insetos, propõe-se a utilização do *few-shot learning* como abordagem. Essa metodologia permite treinar e avaliar o modelo com um número reduzido de exemplos, sendo particularmente valiosa em cenários onde a disponibilidade de imagens é limitada.

O objetivo principal deste trabalho é fornecer uma avaliação detalhada da *Swin Transformer* como uma ferramenta eficaz para a classificação de insetos em imagens agrícolas, especialmente em cenários com um número limitado de imagens disponíveis. Para isso, este trabalho tem como objetivos específicos, analisar qual a melhor versão da *Swin Transformer* para a extração de características de insetos, verificar quão boa essa melhor versão é para classificar apenas insetos na fase adulta em contextos com poucas imagens, e também para classificar apenas insetos em estágio inicial de vida, ainda contendo poucas amostras. Ao fazer isto, este trabalho visa contribuir para o avanço do campo do reconhecimento de insetos em plantações, oferecendo uma avaliação abrangente das diferentes abordagens e seu potencial impacto na agricultura.

1.3 Estrutura do trabalho

Este trabalho é estruturado da seguinte forma: no Capítulo 2, são explicados os conceitos necessários para o entendimento do trabalho, falando sobre o que é a classificação de imagens de insetos e quão importante são os conjuntos de dados para esta tarefa, é feita uma explicação da arquitetura *Swin* e são explicadas as técnicas necessárias para o treinamento de uma rede neural e para a aplicação de *few-shot learning*. No Capítulo 3, são explicados os materiais utilizados no trabalho, além da metodologia dos experimentos realizados, que consistem em verificar qual a melhor versão da *Swin Transformer* para classificação de imagens de insetos e em aplicar *few-shot learning* para imagens de insetos adultos e em estágio inicial de vida. No Capítulo 4 são mostrados os resultados obtidos nos experimentos e são feitas discussões sobre esses resultados e sobre as limitações dos experimentos. Por último, no Capítulo 5, são feitas as conclusões finais do trabalho e são listados próximos passos para possíveis trabalhos futuros.

2 Fundamentação Teórica

2.1 Classificação de Imagens

A identificação de objetos é uma das tarefas mais importantes e desafiadoras da visão computacional bidimensional. Essa tarefa pode ser dividida em algumas subcategorias principais, sendo uma delas a classificação de imagens.

A classificação de imagens consiste em atribuir uma categoria ou rótulo específico a uma imagem, ou região de interesse. O objetivo é identificar o objeto presente na imagem, por exemplo, distinguir entre diferentes insetos. Esse tipo de tarefa é comumente abordado com o uso de algoritmos de aprendizado de máquina, onde o modelo é treinado em um conjunto de dados rotulados ou não para reconhecer e classificar objetos com base em suas características visuais.

2.1.1 Classificação de insetos

A tarefa de classificação de insetos pode ser desafiadora em alguns casos. Muitas das vezes, é preciso classificar os insetos em diferentes estágios da vida, o que torna essa tarefa ainda mais complexa, uma vez que um inseto em fase inicial de vida nada se parece com ele mesmo na fase adulta. Para obter resultados precisos nessa tarefa, é necessário contar com conjuntos de dados extensos para treinar os modelos. A Figura 2.1 ilustra o um exemplo de resultado desse processo, em que foi atribuído um rótulo a uma imagem, correspondente ao inseto presente nesta. Em resumo, uma rede neural é alimentada com uma imagem como entrada e retorna a classe prevista para essa imagem. Durante o treinamento da rede neural, várias imagens são usadas para permitir que o modelo preveja as classes da melhor maneira possível.

Diversos trabalhos, como o Khanramaki, Asli-Ardeh e Kozegar (2021), utilizaram a técnica de classificação de insetos. Nesse trabalho, foram utilizadas redes neurais convolucionais para a classificação de pestes de frutas cítricas. Foi utilizado um conjunto de 1774 imagens juntamente com uma técnica de aumento de dados na fase de treinamento e acurácia dos modelos de RNCs foi medida utilizando validação cruzada de 10 *folds*.

Figura 2.1 – Exemplo de classificação de inseto.

Fonte: autoria própria

O trabalho Wang *et al.* (2020a) utilizou RNCs para a classificação de insetos comuns na agricultura e silvicultura. Foi utilizado um conjunto de originalmente 4909 imagens e também foram utilizadas diversas técnicas de aumento de dados, para ter 73635 imagens no total. Foram obtidos resultados de diferentes RNCs para diferentes taxas de aprendizado, e os resultados de acurácia e tempo de treinamento foram comparados.

Em Li *et al.* (2021b) é feita uma revisão sistemática de classificação de imagens utilizando aprendizado profundo, que mostra todas as etapas deste processo complexo, que vai desde a obtenção das imagens a serem utilizadas, até os testes para verificar o desempenho dos modelos.

2.2 *Dataset*

Os *datasets* são parte fundamental para conseguir altas acurácias nas tarefas de visão computacional. O *dataset* é o conjunto de imagens utilizadas para o treinamento, validação e teste de uma arquitetura de redes neurais. Em tarefas de visão computacional, quanto maior é o *dataset*, melhores costumam ser os resultados obtidos nos testes. O trabalho Deng *et al.* (2009) trouxe inovações para a criação de *datasets*, trazendo o *ImageNet* um *dataset* com aproximadamente 3,2 milhões de imagens de mais de 5 mil classes diferentes. Além do *ImageNet*, outros *datasets* como o *COCO* (Lin *et al.*, 2015) são bastante utilizados para testar a acurácia de novas arquiteturas de redes neurais, como foi feito em Liu *et al.* (2021).

2.2.1 IP102

O IP102, apresentado em Wu *et al.* (2019), é o primeiro conjunto de dados em grande escala criado para a identificação de insetos. O *dataset* contém mais de 75.000 imagens, divididas em 102 classes. Antes do IP102, os *datasets* para o reconhecimento de insetos eram pequenos. Os maiores conjuntos de dados já utilizados para tarefas de reconhecimento de insetos foram em Alfarisy, Chen e Guo (2018), onde foram utilizadas 4511 imagens de 13 classes distintas, em Liu *et al.* (2016), onde foram utilizadas 5100 imagens de 12 classes distintas, e em Xie *et al.* (2018), onde foram usadas 4500 imagens de 40 classes distintas. O conjunto de dados foi construído por meio de pesquisas na internet, que também foram uma fonte fundamental para outros conjuntos de dados, como em Deng *et al.* (2009) e Lin *et al.* (2015). Após a seleção de 120.000 imagens com a colaboração de voluntários para identificação das imagens através dessas pesquisas na internet, especialistas em agricultura foram convidados a analisar as imagens e atribuir as categorias correspondentes, resultando em um total de 75.222 imagens rotuladas.

O conjunto de dados IP102 é composto por duas superclasses, denominadas culturas de campo e culturas econômicas. No total, existem oito culturas: milho, arroz, trigo, beterraba, alfafa, uva, frutas cítricas e manga. Cada uma das 102 classes pertence a um inseto que afeta uma dessas culturas. Existe um significativo desequilíbrio entre as 102 classes presentes no conjunto de dados, onde a classe mais representada possui 5740 amostras, enquanto a classe menos representada possui apenas 71. Esse desequilíbrio, que pode ser visto na Tabela A.10 em que são mostradas as quantidades de imagens de cada uma das classes do *dataset*, pode se tornar um desafio ao tentar classificar as classes menos frequentes no conjunto de imagens. Além disso, o conjunto de dados apresenta outra complicação, uma vez que as classes representam insetos em diferentes estágios de vida. Em outras palavras, uma mesma classe pode conter imagens de uma espécie de inseto em fases distintas, como larva e adulto. Dado que as características das espécies podem variar significativamente em diferentes estágios de vida, essa combinação de vários estágios na mesma classe pode impactar negativamente o desempenho de modelos que visam trabalhar com esse conjunto de dados.

Em termos dos resultados obtidos usando o conjunto de dados no artigo base, para a classificação de insetos, o modelo ResNet obteve a maior precisão, com 49,4%. Para a detecção de insetos, apenas 18983 imagens estão rotuladas, e a YOLOv3 (Redmon; Farhadi, 2018) com a Darknet como espinha dorsal alcançou uma precisão média de 25,67%.

2.3 *Transformers* e a Arquitetura *Swin*

O *Transformer* é uma arquitetura de rede neural profunda que foi proposta em Vaswani *et al.* (2017) que tem sido revolucionária no campo da inteligência artificial, es-

pecialmente no processamento de linguagem natural (PLN). É uma arquitetura usada em modelos sequência-a-sequência, composta por dois componentes principais: o codificador (*encoder*) e o decodificador (*decoder*).

No *Transformer*, o *encoder* recebe uma sequência de entrada e a codifica em uma representação latente usando mecanismos de atenção. A atenção permite que o codificador observe todos os *tokens* da sequência de entrada ao calcular sua representação codificada, capturando as dependências entre esses *tokens* de maneira mais eficiente do que as redes neurais recorrentes tradicionais.

Em seguida, o decodificador, com base na representação latente gerada pelo codificador, gera uma sequência de saída, um *token* de cada vez, usando atenção em sua própria entrada e nas informações contextuais do codificador.

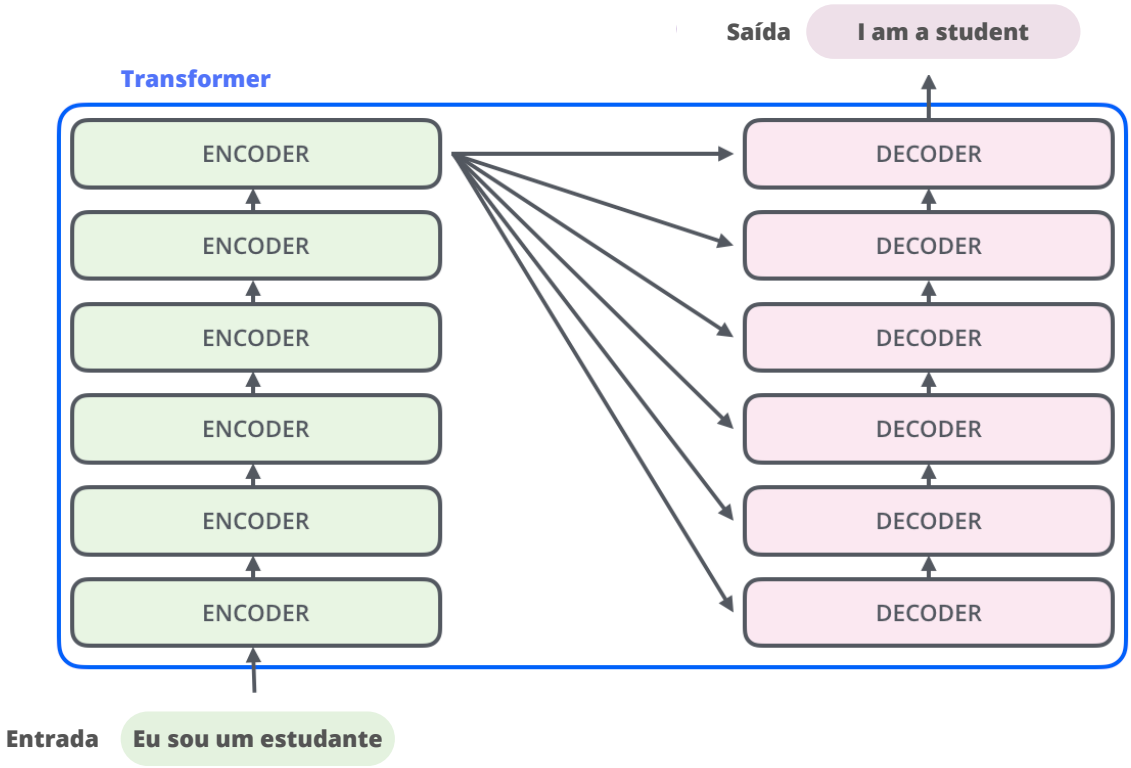

Figura 2.2 – Estrutura de um *Transformer*.

Fonte: adaptado de JAY ALAMMAR (2018, 24)

Na Figura 2.2, é possível ver a estrutura de um *Transformer* que traduz textos do português para o inglês. Por ser uma arquitetura de rede neural profunda, ela é composta por várias camadas de *encoders* e várias camadas de *decoders*. Essas abundâncias de camadas são importantes, pois conseguem agregar informações do início até o fim, conseguindo gerar representações mais contextualizadas da entrada.

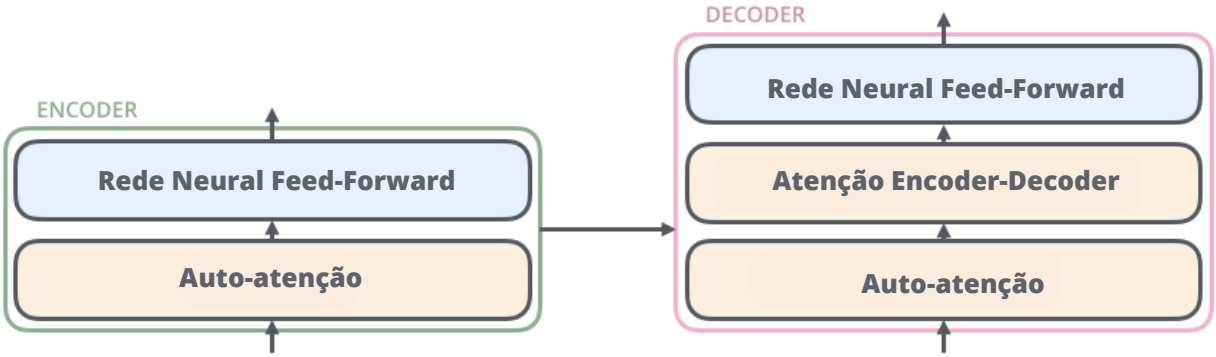

Figura 2.3 – Estrutura das camadas de *encoders* e *decoders*.

Fonte: adaptado de JAY ALAMMAR (2018, 24)

Na Figura 2.3 é possível observar a estrutura de uma camada do encoder, composta por duas camadas, uma camada de autoatenção multi-cabeça (MSA), que permite ao *encoder* "olhar" para os outros *tokens* da entrada quando está codificando um *token* específico da entrada. A outra é uma camada de rede neural *feed-forward* que realiza transformações lineares seguidas de funções de ativação não lineares. É possível também observar que a estrutura do *decoder* é parecida, porém, tem uma camada a mais de atenção *encoder-decoder*, que ajuda o *decoder* a focar nas partes relevantes da entrada.

No contexto de PLN, os *tokens* são geralmente definidos como as palavras individuais de um texto. No entanto, em visão computacional, a definição de *token* é mais complexa. Quando lidamos com uma imagem como entrada, um *token* pode ser entendido como uma parte discreta da imagem, como um *pixel* individual ou um bloco de *pixels*. Cada *token* contém informações específicas da imagem, como cor, textura ou formas. Essa abordagem permite a representação e processamento eficiente de informações visuais por modelos de aprendizado de máquina.

2.3.1 Mecanismos de Atenção

Os mecanismos de atenção, que são parte fundamental da arquitetura *Transformer*, são utilizados para a rede neural conseguir direcionar sua atenção para as partes relevantes da entrada. No contexto de visão computacional, isso é crucial para lidar com a complexidade das imagens, pois permite que a rede se concentre apenas nas regiões ou características mais importantes para a tarefa em questão.

Ao aplicar mecanismos de atenção em visão computacional, a rede neural pode aprender a identificar objetos, padrões ou detalhes significativos em uma imagem, destacando-os com maior ênfase durante o processamento. Isso é especialmente útil em tarefas como detecção de objetos e classificação, em que é necessário distinguir características específicas

dentro da imagem.

Por outro lado, os mecanismos de autoatenção, também conhecidos como atenção de múltiplas cabeças (*multi-head attention*), são uma variação específica dos mecanismos de atenção que têm sido amplamente utilizados na arquitetura *Transformer*. Esses mecanismos de autoatenção são especialmente úteis para capturar relações espaciais e contextuais em imagens, permitindo que a rede neural estabeleça conexões entre diferentes partes da imagem e extraia informações relevantes para a tarefa em questão.

Sempre que um *token*, ou seja, uma parte da imagem, está sendo codificado pelo *encoder*, o mecanismo de autoatenção é acionado para capturar a relação desse *token* com os outros *tokens* da imagem, permitindo uma melhor contextualização da imagem em sua totalidade.

A atenção tem sido amplamente investigada e aplicada no campo da visão computacional, demonstrando seu potencial para aprimorar o desempenho de diversas tarefas. No contexto desta pesquisa, destaca-se o trabalho Hu *et al.* (2019), que introduziu a primeira estrutura visual totalmente baseada em atenção. Essa contribuição pioneira abriu caminho para o desenvolvimento de outras arquiteturas que utilizam atenção no campo da visão computacional, rompendo com o domínio tradicional das redes neurais convolucionais (RNCs).

Há diversos desafios na adaptação de modelos *Transformers* da linguagem para a visão, devido às diferenças entre os dois domínios. Isso inclui grandes variações na escala das imagens e a alta resolução de *pixels* presentes nessas imagens, em comparação com as palavras nos textos.

A primeira arquitetura *Transformer* projetada especificamente para aplicações de visão computacional foi a *Vision Transformer* (*ViT*), apresentada em Dosovitskiy *et al.* (2021). Esse trabalho demonstrou que a arquitetura *ViT* é capaz de alcançar resultados comparáveis aos obtidos pelos modelos de ponta baseados em RNCs.

No modelo *ViT*, as imagens são decompostas em *patches* de 16×16 *pixels*, que são transformados em vetores de *patches* por meio de uma transformação linear. Esses vetores de *patches* são combinados com incorporações posicionais antes de serem processados pelo *Transformer*.

Um grande problema da *ViT* é que os *patches* de 16×16 *pixels* podem ser muito grandes, o que pode ser problemático para tarefas que exigem processamento detalhado de cada *pixel*. Por exemplo, se uma imagem *Full HD* de 1920×1080 *pixels* fosse ser usada como entrada do *ViT*, e cada *pixel* fosse um *token*, mais de dois milhões de *tokens* seriam necessários apenas para uma única imagem, o que torna a tarefa não escalável. A complexidade computacional aumenta de forma quadrática com relação a uma dimensão da imagem, o que torna o método inviável em termos de escalabilidade.

Devido a essa limitação, uma série de pesquisas e desenvolvimentos subsequentes foram impulsionados no campo da visão computacional, explorando o potencial da *Vision Transformer* para tarefas de processamento de imagens. Nesse contexto, foi criada a *Swin Transformer*, visando resolver os problemas encontrados pela *ViT*.

2.3.2 Swin Transformer

A *Swin Transformer* é uma arquitetura *Transformer* hierárquica que utiliza janelas deslocadas para processar eficientemente imagens de alta resolução apresentada em Liu *et al.* (2021). Em vez de dividir a imagem em *patches* com uma janela fixa, a abordagem de janela deslocada adotada pela *Swin Transformer* desloca cada *patch* em uma determinada direção em relação aos *patches* vizinhos. Isso permite capturar informações contextuais globais e reduzir a perda de informações devido à sobreposição excessiva entre *patches* adjacentes. Além disso, uma grande inovação dessa arquitetura com relação à *ViT*, é que a complexidade computacional desse modelo aumenta de forma linear com relação a uma dimensão da imagem, e não quadrática, facilitando o processamento de imagens de alta resolução. Essa arquitetura desempenha um papel competente como uma estrutura principal de propósito geral para visão computacional.

Essa arquitetura vem sendo utilizada em diversos trabalhos recentes, para diferentes propósitos, como restauração de imagens (Liang *et al.*, 2021), segmentação de imagens médicas (Lin *et al.*, 2022), reconhecimento de expressões faciais (Kim; Kim; Won, 2022), detecção de veículos para controle de tráfego (Deshmukh *et al.*, 2023) e até mesmo contagem de pessoas em multidões (Liang *et al.*, 2022).

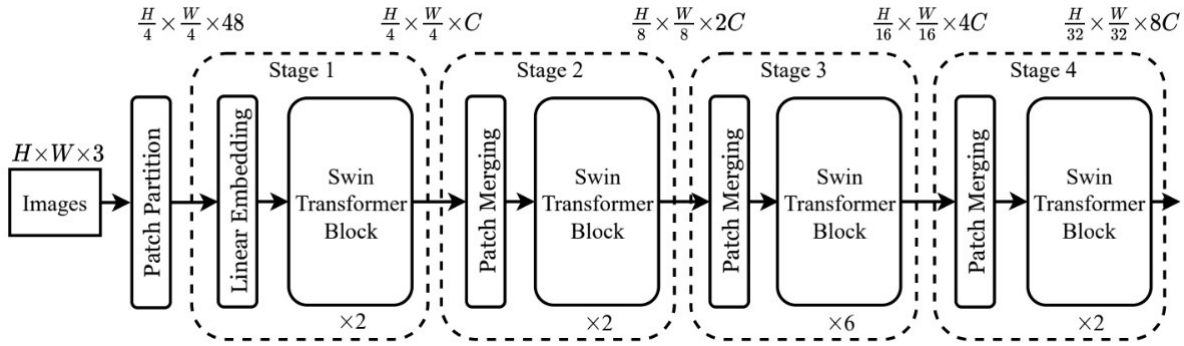

Figura 2.4 – A arquitetura da *Swin Transformer* (Swin-T).

Fonte: Liu *et al.* (2021)

Na Figura 2.4 é mostrada uma ilustração da arquitetura da versão *tiny* da *Swin Transformer*, em que é possível ver 4 tipos diferentes de camadas.

a) *Patch Partition*

Nesta primeira camada, a imagem *RGB* de entrada é divida em *patches* menores e não sobrepostos. Esses *patches* são os *tokens*, representados como uma concatenação dos valores RGB dos *pixels*. Nesta arquitetura, cada um dos *tokens* possui dimensão de 4×4 *pixels*, e considerando que cada *token* possui três canais de cor (RGB), tem-se que a dimensão de cada *token* é de $4 \times 4 \times 3 = 48$.

b) *Linear Embedding*

Nesta camada, ilustrada na Figura 2.5, uma camada de incorporação linear (*linear embedding*) é aplicada nos *tokens* para ter um resultado com uma dimensão arbitrária C. As variantes da arquitetura *Swin* possuem diferentes valores de C, que denotam a capacidade do modelo.

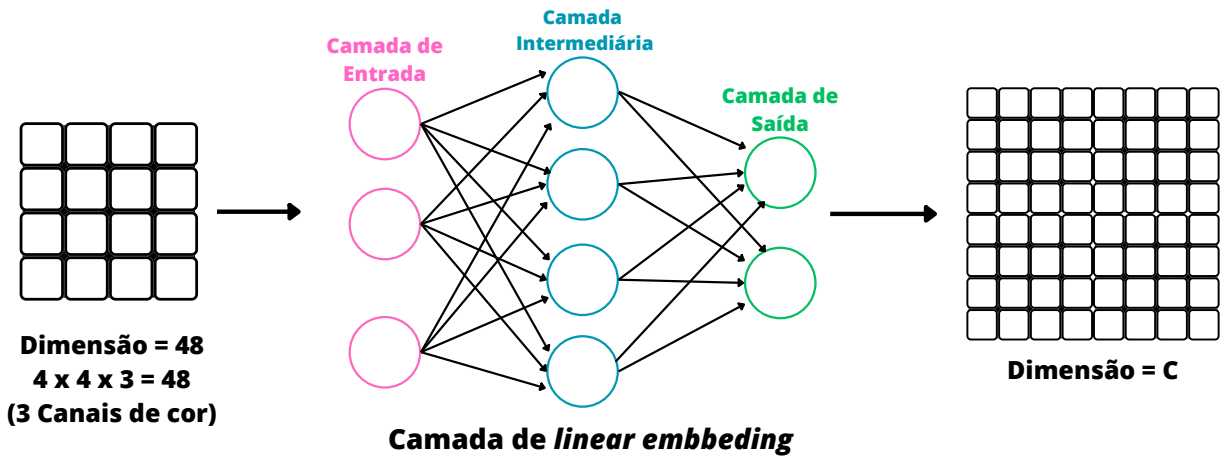

Figura 2.5 – Exemplo de como funciona a camada de *Linear Embedding*.

Fonte: autoria própria.

c) *Patch Merging*

O *patch merging* é uma técnica de *downsampling* livre de convolução que concatena as características de cada grupo de patches vizinhos $N \times N$. Para produzir a representação hierárquica do modelo, o número de *tokens* é reduzido por um fator N nestas camadas de *patch merging*. Conforme a rede vai se aprofundando, menor é o número de *tokens*. Na primeira camada de *patch merging* no estágio 2 da Figura 2.4, são concatenadas as características de cada grupo de 2×2 *patches* vizinhas, que reduz o número de *tokens* por 4. Após isso uma camada linear (*Layer Norm*) é aplicada no vetor de dimensão *4C*, reduzindo para dimensão *2C*, conforme pode ser visto na Figura 2.6. A resolução final após esta etapa é de $\frac{H}{8} \times \frac{W}{8}$. Esta camada também é aplicada nos estágios 3 e 4, com as resoluções da saída sendo respectivamente $\frac{H}{16} \times \frac{W}{16}$ e $\frac{H}{32} \times \frac{W}{32}$.

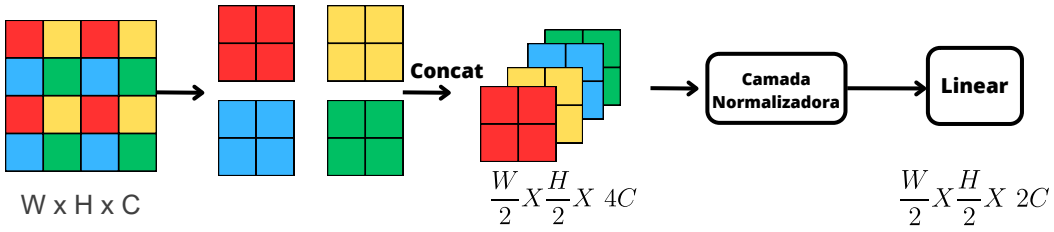

Figura 2.6 – Exemplo de como funciona a camada de *Patch merging*.

Fonte: autoria própria.

d) *Swin Transformer Block*

Por último, a camada de *Swin Transformer Block* é o diferencial da arquitetura, que é onde a atenção é computada com as janelas deslocadas.

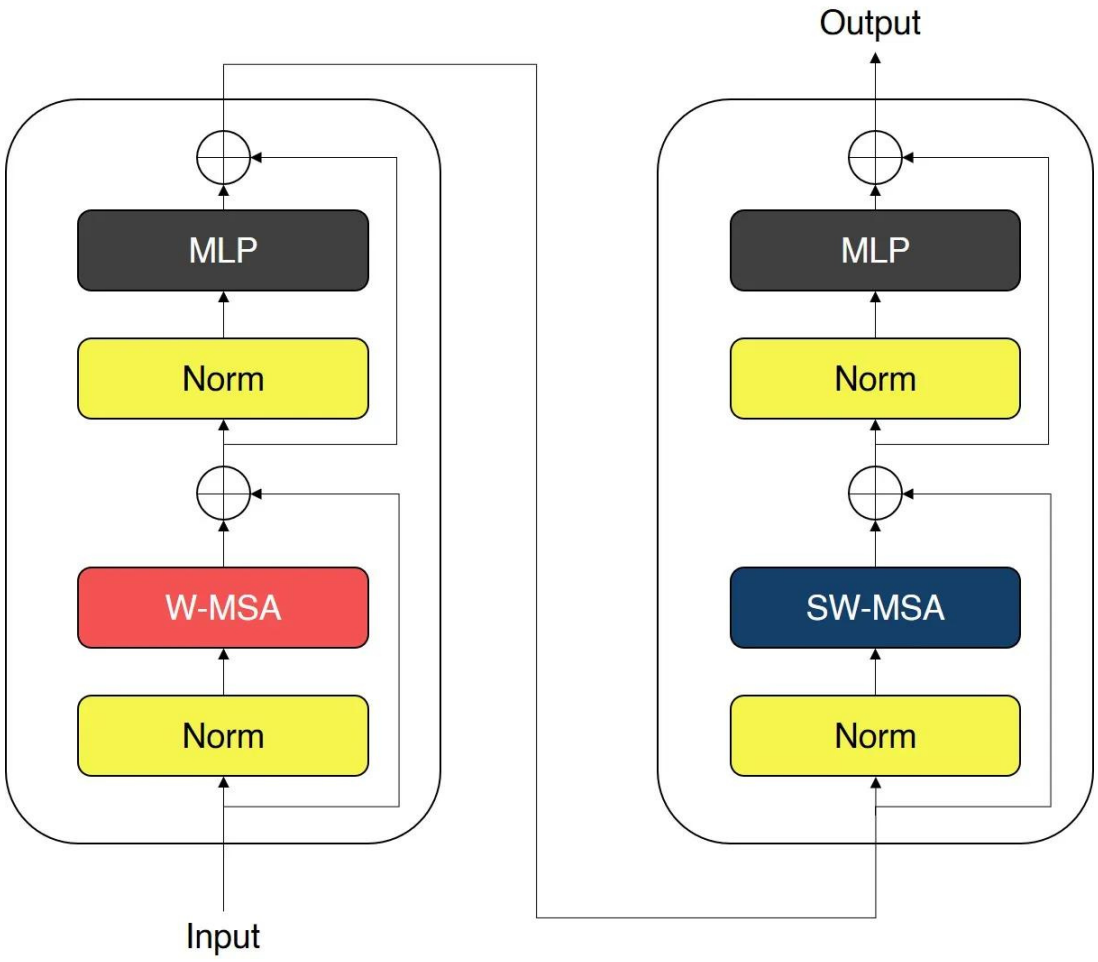

Figura 2.7 – Exemplo de como funciona a camada de dois blocos consecutivos de *Swin Transformer*.

Fonte: JAMES LOY (2022)

Um bloco *Swin Transformer* é composto por diversos módulos como pode ser visto na Figura 2.7. O *MSA* padrão, que foi utilizado no *ViT*, computa a atenção de um *token* em relação a todos os outros *tokens* da imagem, o que resulta em uma complexidade quadrática com relação ao número de *tokens*, o que torna o *ViT* inutilizável para imagens de alta resolução. Dessa forma, na arquitetura *Swin* foram propostas diferentes *MSAs*, baseados em janelas.

– *W-MSA*

  O *W-MSA* é um módulo de autoatenção multi-cabeça baseado em janelas. Uma janela é uma coleção de *tokens* e a atenção neste módulo só é computada entre os *tokens* que fazem parte da janela, ou seja, todo *token* em uma janela específica é computado com relação a todos os outros *tokens* dessa janela.

  Com o tamanho de janelas fixado conforme o aprofundamento da rede, a complexidade computacional do *W-MSA* é linear em relação ao número de *tokens*, ou seja, ao tamanho da imagem, o que representa uma grande melhora com relação ao *MSA* padrão.

– *SW-MSA*

  A utilização de *W-MSA* apesar de apresentar uma grande melhora na complexidade computacional, limita o poder de modelagem da rede, uma vez que a atenção só é computada dentro das janelas. Diante disso, a *Swin Transformer* além de utilizar a *W-MSA*, utiliza a *SW-MSA*, que é um módulo de autoatenção multi-cabeças baseado em janelas deslocadas utilizado logo após o módulo *W-MSA* na rede.

  Para estabelecer conexões entre as janelas, o módulo *SW-MSA* realiza um deslocamento das janelas em direção ao canto inferior direito por um fator de M/2, onde M é o tamanho da janela, que por padrão é 7 *pixels*. Esse deslocamento visa criar conexões cruzadas entre as janelas adjacentes.

  No entanto, essa mudança acarreta em "*patches* órfãos" que não pertencem a nenhuma janela e janelas com *patches* incompletos. Para solucionar essa questão, a *Swin Transformer* emprega a técnica de deslocamento cíclico, movendo os "*patches* órfãos" para as janelas com *patches* incompletos. Após esse deslocamento, é importante destacar que uma janela pode conter *patches* que não são adjacentes no mapa de características original.

  Portanto, durante o cálculo da autoatenção, é aplicada uma máscara para limitar a consideração apenas aos *patches* adjacentes. Essa abordagem de janelas deslocadas, combinada com o uso da técnica de deslocamento cíclico, permite a criação de conexões significativas entre as janelas, resultando em um melhor desempenho geral da rede.

Além dos módulos *MSA*, a camada também é composta por camadas de normalização (*Norm*) por perceptrons multicamadas (*MLP*).

Com essa arquitetura, a *Swin Transformer* é capaz de processar imagens de alta resolução com grande desempenho devido ao seu custo computacional linear a dimensão da imagem. A *Swin Transformer* possui diferentes variantes com complexidades computacionais diferentes:

a) *Swin-T*: a *Swin Tiny* possui a dimensão $C = 96$, e o número de camadas para cada estágio, referente à Figura 2.4 é de 2,2,6,2.

b) *Swin-S*: a *Swin Small* possui o dobro do tamanho do modelo *Swin-T*, além do dobro de complexidade computacional, e possui $C = 96$ e o número de camadas igual à 2,2,18,2.

c) *Swin-B*: a *Swin Base* possui o dobro do tamanho do modelo *Swin-S*, além do dobro de complexidade computacional, e possui $C = 128$ e o número de camadas igual à 2,2,18,2.

d) *Swin-L*: a *Swin Large* possui o dobro do tamanho do modelo *Swin-B*, além do dobro de complexidade computacional, e possui $C = 192$ e o número de camadas igual à 2,2,18,2.

No artigo base, a *Swin Transformer* teve seu desempenho comparado aos modelos RegNet (Radosavovic *et al.*, 2020), *ViT* (Dosovitskiy *et al.*, 2021) e *DeiT*(Touvron *et al.*, 2021) no ImageNet, e conseguiu obter um resultado melhor que todas essas acurácias na tarefa de classificação, obtendo um máximo de 87,3% com a *Swin-L* no ImageNet 22k, se mostrando uma ótima opção para esta tarefa.

2.3.2.1 *Swin Transformer V2*

A *Swin Transformer V2*, apresentada em Liu *et al.* (2022), é uma versão aprimorada da arquitetura *Swin Transformer* (Liu *et al.*, 2021), que alcançou resultados impressionantes em várias tarefas de visão computacional. A principal diferença da versão V2 é sua capacidade de lidar com imagens de alta resolução e uma quantidade maior de parâmetros.

Com essa escalabilidade, a *Swin Transformer V2* estabeleceu novos recordes em *benchmarks* de visão computacional importantes. Por exemplo, obteve uma alta precisão de classificação no *ImageNet-V2* (84,1%) e excelentes resultados em detecção de objetos na *COCO*.

A *Swin Transformer V2* superou desafios de instabilidade durante o treinamento e transferência eficaz de modelos pré-treinados em baixas resoluções para resoluções mais altas. Foram introduzidas técnicas inovadoras, como pós-normalização residual e atenção

cosseno escalada, além de uma abordagem de viés posicional contínuo espaçado em escala logarítmica.

Essas melhorias permitiram economia significativa de memória da GPU e tornaram o treinamento de modelos de visão de grande escala possível em GPUs convencionais. Através do pré-treinamento auto-supervisionado, a *Swin Transformer V2* alcançou um desempenho de ponta em várias tarefas de visão com alta resolução.

A *Swin Transformer V2* representa um avanço significativo na capacidade de lidar com imagens de alta resolução e estabelece novos padrões de desempenho em diversas tarefas de visão computacional em comparação com a versão original.

2.4 Treinamento de uma rede neural

O treinamento de uma rede neural em tarefas de visão computacional, envolve um conjunto complexo de etapas que desempenham um papel crucial no desenvolvimento e no aprimoramento do modelo. Ele visa fazer com que a rede neural consiga ser eficaz em uma determinada tarefa, com a classificação ou detecção de insetos em plantações. Neste contexto, a aprendizagem supervisionada é uma das abordagens mais comuns utilizadas para o treinamento de redes neurais em visão computacional.

2.4.1 Aprendizagem supervisionada

A aprendizagem supervisionada é um método de aprendizado de máquina em que um modelo é treinado para aprender uma função a partir de exemplos de entrada e suas respectivas saídas rotuladas. No contexto da visão computacional, cada imagem utilizada para treinar a rede é associada a uma classe específica, permitindo que o modelo aprenda a reconhecer características visuais correspondentes a cada classe.

Embora existam outros tipos de aprendizagem, como a não supervisionada e a mista, a aprendizagem supervisionada é a abordagem mais adequada para tarefas de visão computacional. Isso se deve ao fato de que a aprendizagem supervisionada utiliza dados rotulados, proporcionando ao modelo uma compreensão mais precisa e confiável das imagens. Ao aprender com exemplos rotulados, o modelo pode generalizar e classificar novas imagens de forma mais eficaz.

Em Mahadevkar *et al.* (2022) foi feita uma revisão com diferentes tipos de aprendizagem no campo da visão computacional, e foi concluído que a aprendizagem supervisionada é uma das melhores técnicas no campo, conseguindo atingir acurácias de até 99% nos testes feitos, porém sofre com alguns desafios como a qualidade ruim das imagens presentes nos conjuntos de dados, além de necessitar de muitos dados para a etapa de treinamento para conseguir atingir essas altas acurácias.

2.4.2 Separação de Dados

No processo de aprendizagem supervisionada, é comum dividir as etapas em treinamento, validação e teste. É essencial garantir que as imagens usadas em cada uma dessas etapas sejam diferentes, a fim de obter um resultado não enviesado. A maioria dos dados é destinada ao conjunto de treinamento, enquanto uma pequena porção é separada para as etapas de validação e teste.

Durante a etapa de treinamento, as imagens do conjunto de treinamento são utilizadas para o modelo aprender, ajustando seus parâmetros internos e otimizando seu desempenho. O objetivo principal é permitir que o modelo generalize a partir dos dados de treinamento, para realizar previsões precisas em novos dados.

Durante a etapa de validação, o modelo faz previsões nas imagens do conjunto de validação, cujos rótulos verdadeiros são conhecidos. Os resultados obtidos são comparados com os rótulos verdadeiros para medir o desempenho do modelo. Essa etapa permite ajustar os hiperparâmetros do modelo, como taxa de aprendizado e arquitetura da rede neural, visando melhorar sua capacidade de generalização.

Por fim, após o treinamento e a validação, o desempenho final do modelo é avaliado usando o conjunto de teste. Esse conjunto de teste consiste em imagens que o modelo nunca viu antes, representando dados totalmente novos para ele. Os resultados obtidos nesse conjunto de teste fornecem uma estimativa realista do desempenho esperado do modelo em aplicações reais.

Dessa forma, a divisão adequada das imagens entre as etapas de treinamento, validação e teste é crucial para evitar problemas de viés e sobreajuste, garantindo que o modelo seja capaz de generalizar corretamente. Ao reservar dados independentes para a etapa de teste, é possível realizar uma avaliação objetiva do desempenho do modelo e assegurar sua eficácia em cenários do mundo real. Vale ressaltar que a etapa de validação, embora seja comumente utilizada, não é obrigatória e pode ser omitida em certos casos, dependendo das restrições do conjunto de dados e dos recursos disponíveis. Uma separação comumente feita quando há apenas dados treino e teste, é de 80% para treino e 20% para teste (Mohri; Rostamizadeh; Talwalkar, 2018). No entanto, é importante enfatizar a importância de reservar dados independentes para o teste, a fim de avaliar a capacidade de generalização do modelo e garantir sua confiabilidade em situações reais.

2.4.3 *Fine-tuning*

O *fine-tuning* é uma técnica utilizada para aproveitar modelos pré-treinados em grandes conjuntos de dados, como o *ImageNet* (Deng *et al.* (2009)). Para treinar modelos nesses grandes conjuntos de dados, são necessários recursos computacionais significativos, além de bastante tempo. O *fine-tuning* permite adaptar esses modelos pré-treinados e seus

pesos para um novo conjunto de dados ou tarefa.

Nesse processo, o modelo pré-treinado é inicializado com seus pesos já aprendidos e, em seguida, continua o treinamento com um novo conjunto de dados específico. Durante o *fine-tuning*, os pesos do modelo são ajustados para aprender representações mais especializadas para o novo conjunto de dados, enquanto as informações aprendidas anteriormente são mantidas.

Normalmente, apenas as camadas finais do modelo são modificadas durante o *fine-tuning*, mantendo as camadas iniciais, responsáveis por capturar características gerais, fixas. Essa abordagem permite que o modelo se adapte ao novo conjunto de dados, aproveitando as informações prévias e acelerando o processo de treinamento. O *fine-tuning* é uma estratégia eficaz para utilizar o conhecimento prévio dos modelos pré-treinados em novos cenários, permitindo um treinamento mais rápido e eficiente.

2.4.4 Validação Cruzada

Em uma situação em que existem apenas conjuntos de treinamento e teste, é comum que a aleatoriedade na divisão dos dados em 80% para treinamento e 20% para teste possa resultar em variações significativas nos resultados ao executar o modelo várias vezes. No entanto, para lidar com esse problema, uma abordagem recomendada é a utilização da técnica de validação cruzada.

2.4.4.1 Validação Cruzada *k-fold*

A técnica de validação cruzada *k-fold* consiste em dividir o conjunto de dados em k partições (*folds*) e executar o modelo k vezes, em que cada uma das vezes uma das partições é utilizada para teste, e as outras $k - 1$ partições são utilizadas para treinamento.

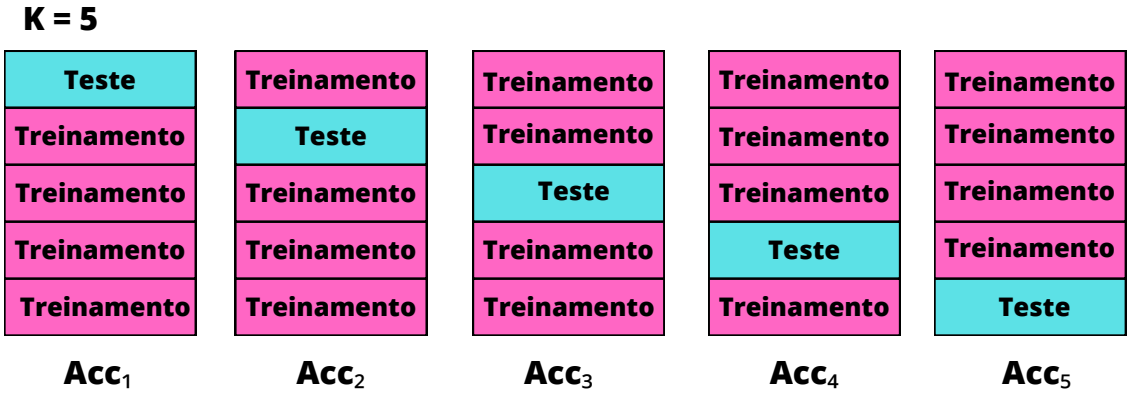

Figura 2.8 – Validação cruzada *k-fold* para $k = 5$.

Fonte: autoria própria.

A utilização da técnica de validação cruzada permite mitigar a dependência de uma única divisão aleatória dos dados, proporcionando uma avaliação mais estável e confiável do modelo. Essa abordagem desempenha um papel fundamental ao identificar a consistência do desempenho do modelo ou se este varia significativamente diante de diferentes conjuntos de dados de treinamento e teste. Na Figura 2.8 é possível ver um exemplo de separação dos dados utilizando validação cruzada *5-fold*.

A acurácia de um modelo utilizando a validação cruzada *k-fold* pode ser obtida pela média das acurácias obtidas em cada uma das partições, ou seja, considerando que Acc_i é a acurácia obtida pelo modelo no i-ésimo *fold*, tem-se que:

$$Acc_{modelo} = \frac{\sum_{i=1}^{k} Acc_i}{k} \tag{2.1}$$

2.4.5 Funções de Ativação

Uma função de ativação é uma função matemática aplicada à saída de um neurônio ou a um conjunto de neurônios em uma rede neural. Ela descreve a relação entre os valores de entrada e os valores de saída de um neurônio ou camada, podendo ser linear ou não-linear. Em Dubey, Singh e Chaudhuri (2022) foi feito um trabalho mostrando as principais funções de ativação comumente utilizadas em treinamentos de redes neurais, mostrando as principais vantagens e desvantagens de cada uma delas.

No caso das funções de ativação lineares, a relação entre os valores de entrada e saída é mantida de forma linear, ou seja, a saída é uma combinação linear dos valores de entrada, multiplicados por um coeficiente e somados a uma constante. Uma desvantagem do uso desse tipo de função é que caso o problema que a rede esteja tentando resolver seja complexo, ou seja, não linear, não importa quantas camadas a rede possui, ela será incapaz de aprender o seu funcionamento. Uma importante função de ativação linear é a ReLU (Função Linear Retificada), que retorna 0 para valores negativos e mantém os valores positivos sem alteração.

Por outro lado, as funções de ativação não-lineares introduzem comportamentos não-lineares na relação entre os valores de entrada e saída. Isso permite que a rede neural aprenda e represente relações mais complexas nos dados. Exemplos de funções de ativação não-lineares são a função sigmoidal, a tangente hiperbólica (tanh), a função *softmax* e diversas variações da ReLU.

A seguir, podem ser vistas algumas destas funções:

a) ReLU: $f(x) = max(0,x)$;

b) Função sigmoidal: $f(x) = \frac{1}{1+e^{-x}}$;

c) Função tangente hiperbólica: $f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$.

2.4.6 Funções de Perda

Uma função de perda é uma função matemática que permite avaliar a discrepância entre as saídas previstas pela rede neural e os resultados esperados. Essas funções desempenham um papel fundamental durante o processo de treinamento, pois quantificam a qualidade do modelo em relação aos dados de treinamento.

Ao fornecer os parâmetros atuais da rede neural, a função de perda calcula um valor que representa a medida de erro ou desajuste entre as previsões do modelo e os rótulos reais dos dados. Durante o treinamento, o objetivo é minimizar o valor dessa função de perda, ajustando os parâmetros do modelo de forma iterativa por meio de algoritmos de otimização.

Existem várias funções de perda disponíveis, cada uma adequada para diferentes tipos de problemas e requisitos do modelo. Alguns exemplos comuns de funções de perda incluem a entropia cruzada binária, a entropia cruzada categórica e o erro quadrático médio.

A entropia cruzada binária (Ruby; Yendapalli, 2020) é frequentemente utilizada em problemas de classificação binária, nos quais o objetivo é atribuir uma única classe a cada exemplo de dados. Essa função de perda mede a discrepância entre as probabilidades previstas pelo modelo e as classes reais dos dados. Seu propósito é minimizar a divergência entre as distribuições de probabilidade, permitindo que o modelo aprenda a realizar classificações mais precisas.

Já a entropia cruzada categórica é empregada em problemas de classificação multiclasse, nos quais existem várias classes possíveis para cada exemplo de dados. Essa função de perda calcula a diferença entre as probabilidades previstas pelo modelo e as distribuições de classes reais. Seu objetivo é minimizar a divergência entre as distribuições de probabilidade, permitindo que o modelo atribua corretamente as probabilidades às diferentes classes.

O erro quadrático médio (MSE) é comumente utilizado em problemas de regressão, nos quais o objetivo é prever um valor numérico contínuo. Essa função de perda mede a média dos quadrados das diferenças entre as previsões do modelo e os valores reais. O MSE penaliza fortemente as discrepâncias maiores, incentivando o modelo a reduzir a magnitude dos erros e ajustar-se aos dados com maior precisão.

A escolha da função de perda depende da natureza do problema em questão e das métricas de desempenho desejadas. É fundamental selecionar uma função de perda apropriada para guiar o processo de treinamento e otimização, permitindo que o modelo aprenda a reduzir o erro e se aproxime cada vez mais da resposta esperada. Dessa forma, o desempenho geral do modelo é aprimorado e sua capacidade preditiva é maximizada.

2.4.7 Retropropagação

Retropropagação é um algoritmo essencial utilizado durante o treinamento de redes neurais (Osman; Blostein, 1999). Ele calcula os gradientes dos parâmetros da rede em relação à função de perda, permitindo que o modelo ajuste seus parâmetros de forma iterativa. A retropropagação ocorre após o cálculo da função de perda, propagando os gradientes de volta pela rede neural. Esses gradientes indicam a direção e magnitude da atualização necessária para minimizar a perda. Em seguida, o otimizador utiliza esses gradientes para atualizar os parâmetros do modelo.

2.4.8 Otimizadores

Otimizadores são algoritmos utilizados durante o treinamento de redes neurais para ajustar os parâmetros do modelo para minimizar a função de perda. Eles determinam a direção e a magnitude das atualizações dos parâmetros com base nos gradientes da função de perda calculados na etapa de retropropagação em relação aos parâmetros. Dois exemplos populares de otimizadores são o Gradiente Descendente Estocástico (SGD) (Tian; Zhang; Zhang, 2023) e a Estimativa de Momento Adaptável (Adam) (Kingma; Ba, 2014).

O SGD é um otimizador básico que atualiza os parâmetros em pequenos incrementos a cada iteração do treinamento. Ele utiliza uma taxa de aprendizado fixa para determinar o tamanho dos passos de atualização. Durante o treinamento, o SGD seleciona um subconjunto aleatório dos dados de treinamento, chamado de *mini-batch*, e utiliza a retropropagação para calcular os gradientes da função de perda em relação aos parâmetros. Em seguida, os parâmetros são atualizados na direção oposta ao gradiente, ajustando-se para minimizar o valor da função de perda. O SGD é computacionalmente eficiente e fácil de implementar, sendo amplamente utilizado em muitas aplicações.

O Adam é um otimizador avançado que combina adaptação de taxa de aprendizado e momentos de gradiente. Ele mantém estimativas adaptativas do momento de primeira ordem, que corresponde à média dos gradientes, e do momento de segunda ordem, que corresponde à média das magnitudes dos gradientes ao quadrado, para cada parâmetro. Durante o treinamento, essas estimativas são atualizadas e utilizadas para calcular os passos de atualização dos parâmetros. O Adam ajusta automaticamente a taxa de aprendizado com base nas estimativas de momento, adaptando-se a diferentes taxas de aprendizado para diferentes parâmetros. Essa abordagem aprimora a convergência e a eficiência do treinamento, o que torna o Adam uma escolha popular em muitas aplicações de redes neurais.

Tanto o SGD quanto o Adam são otimizadores amplamente utilizados na otimização de redes neurais. A escolha entre eles depende das características do problema, do tamanho do conjunto de dados, da arquitetura da rede neural e das preferências do desenvolvedor.

Ambos desempenham um papel essencial no ajuste dos parâmetros do modelo durante o treinamento, permitindo que a rede neural se adapte e melhore seu desempenho em relação à função de perda.

2.4.9 Taxa de Aprendizado

A taxa de aprendizado é um hiperparâmetro fundamental nos algoritmos de otimização empregados no treinamento de redes neurais. Essa taxa governa o tamanho dos incrementos aplicados aos parâmetros do modelo durante o processo de atualização iterativa (Hamed; Desouky, 1996).

A taxa de aprendizado exerce controle sobre a velocidade com que o modelo se ajusta aos dados de treinamento. Valores mais elevados de taxa de aprendizado podem gerar atualizações maiores nos parâmetros, levando a uma convergência mais rápida, porém, potencialmente menos estável. Por outro lado, valores mais baixos de taxa de aprendizado resultam em incrementos menores nos parâmetros, o que torna a convergência mais lenta, embora possa proporcionar maior precisão e estabilidade.

A seleção apropriada da taxa de aprendizado é essencial para obter um desempenho satisfatório do modelo. Uma taxa de aprendizado excessivamente alta pode impedir que o processo de treinamento convirja ou causar oscilações em torno de mínimos globais, dificultando a obtenção de um modelo bem ajustado. Por outro lado, uma taxa de aprendizado excessivamente baixa pode levar a um treinamento lento e resultar em um modelo que não converge ou que converge para mínimos locais subótimos.

A escolha da taxa de aprendizado adequada é complexa e depende de diversos fatores, como a complexidade do problema, a quantidade e qualidade dos dados, a arquitetura da rede neural e o otimizador utilizado. Durante o treinamento, é comum ajustar a taxa de aprendizado por meio de técnicas como decaimento da taxa de aprendizado ou otimização adaptativa. Essas estratégias visam alcançar um equilíbrio entre a obtenção de uma convergência rápida e a estabilidade do modelo.

2.4.10 *Batches*

Os *batches* são subconjuntos dos dados de treinamento utilizados para estimar os gradientes e atualizar os parâmetros do modelo. Em vez de processar todos os dados de uma vez, eles são processados em pequenos lotes, que são os *batches*. Essa abordagem traz várias influências importantes no treinamento. Em Radiuk (2017) é feita uma análise do impacto do tamanho desses conjuntos em treinamentos de RNCs para diversos datasets, e mostra que é um parâmetro crucial para obter boas acurácias em tarefas de reconhecimento de imagens.

Primeiramente, o uso de *batches* melhora a eficiência computacional do treinamento. Ao calcular os gradientes para lotes menores de dados, reduz-se a carga computacional

e o consumo de memória. Isso torna o treinamento mais viável em termos de recursos, especialmente quando se trabalha com conjuntos de dados grandes.

Além disso, os *batches* têm um impacto na regularização e na generalização do modelo. Ao atualizar os parâmetros com base em lotes menores de dados, evita-se o superajuste, no qual o modelo se ajusta excessivamente aos padrões específicos dos exemplos individuais. Com a abordagem de *batches*, o modelo tem uma melhor capacidade de generalização para dados não vistos, melhorando sua capacidade de lidar com novos exemplos.

Por fim, o uso de *batches* permite uma melhor paralelização do treinamento. Em sistemas com múltiplas unidades de processamento, como GPUs, cada lote pode ser processado em paralelo. Isso acelera o tempo de treinamento e permite uma utilização mais eficiente dos recursos computacionais disponíveis.

Em resumo, o uso de *batches* é uma estratégia fundamental para o treinamento eficiente e estável de redes neurais. A escolha do tamanho do *batch* é um hiperparâmetro importante a ser ajustado conforme as características do problema e dos recursos disponíveis. Um *batch* muito pequeno pode gerar uma estimativa do gradiente muito variável, dificultando a convergência do modelo, além de ser um uso ineficiente de recursos computacionais. Por outro lado, utilizar um *batch* muito grande pode nem ser possível devido às limitações de *hardware*, além de resultar em uma menor variedade de exemplos sendo apresentados ao modelo a cada atualização de parâmetros, prejudicando sua capacidade de generalização. Portanto, é necessário encontrar um equilíbrio adequado para o tamanho do *batch*, levando em consideração tanto a estabilidade do treinamento quanto as restrições de recursos.

2.4.11 Épocas

As épocas, no contexto de treinamento de redes neurais, referem-se a uma unidade de iteração durante o processo de treinamento. Uma época é definida como um ciclo completo em que todos os exemplos do conjunto de treinamento passam pela rede neural.

Durante uma época, todos os exemplos de treinamento são alimentados à rede neural, e seus resultados são comparados com as saídas desejadas. Com base nessas comparações, os parâmetros do modelo são atualizados usando um algoritmo de otimização a fim de minimizar a função de perda. Quando a técnica de validação também é utilizada durante o treinamento, ela ocorre ao final de cada época.

O número de épocas é um hiperparâmetro que precisa ser definido antes do treinamento. Geralmente, quanto maior o número de épocas, mais oportunidades o modelo tem de aprender e ajustar seus parâmetros em relação aos dados de treinamento. No entanto, o uso de um número excessivo de épocas pode levar ao sobreajuste (*overfitting*), onde o modelo se ajusta muito bem aos dados de treinamento, mas não generaliza bem para novos dados.

Em Afaq e Rao (2020) foi feito um estudo mostrando a significância desse hiperpa-

râmetro no treinamento de uma rede neural. A escolha do número adequado de épocas depende da complexidade do problema, do tamanho do conjunto de treinamento e da taxa de convergência do modelo. É comum monitorar as métricas de desempenho do modelo, como a perda ou a acurácia, ao longo das épocas e interromper o treinamento quando essas métricas atingem um ponto de saturação ou começam a se deteriorar.

Em resumo, uma época representa um ciclo completo em que todos os exemplos de treinamento são apresentados à rede neural e seus parâmetros são atualizados. O número de épocas é um hiperparâmetro importante a ser definido e pode afetar o desempenho e a capacidade de generalização do modelo.

2.4.12 Métricas de Avaliação do Modelo

Existem diversas formas de avaliar o desempenho de um modelo após o seu treinamento. Além da acurácia, que mede a proporção de previsões corretas em relação ao total de exemplos, existem outras métricas e técnicas disponíveis para realizar essa avaliação de forma abrangente. A matriz de confusão, por exemplo, é uma ferramenta valiosa que apresenta as classificações corretas e os erros, permitindo uma análise mais detalhada do comportamento do modelo em cada classe. A partir da matriz de confusão, é possível calcular métricas adicionais, como precisão, *recall* e *F1-score*, que fornecem informações específicas sobre o desempenho em cada classe.

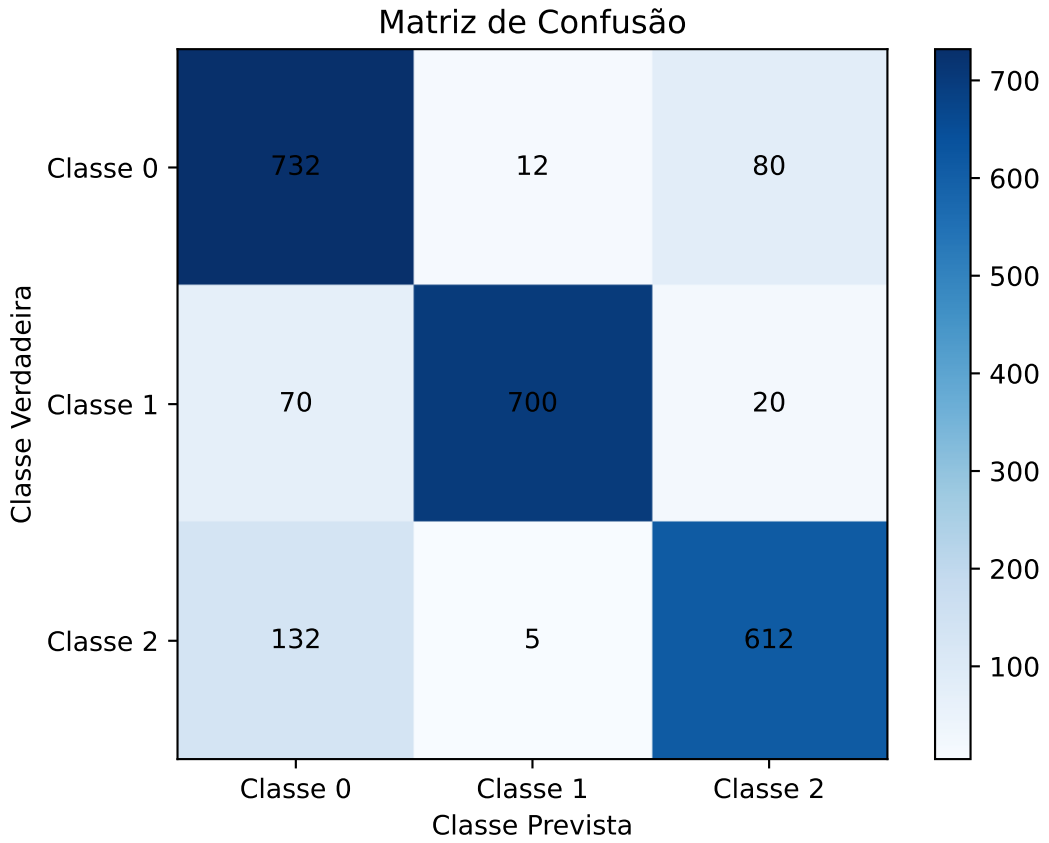

Figura 2.9 – Exemplo de matriz de confusão em modelo com 3 classes.

Fonte: autoria própria.

Na Figura 2.9, é possível ver um exemplo de uma matriz de confusão, que mostra todas as previsões e as verdadeiras classes de todos os dados. A partir dessa matriz, é possível calcular os valores de falsos positivos, falsos negativos, verdadeiros positivos e verdadeiros negativos de cada classe e assim calcular as métricas adicionais mencionadas anteriormente. A Figura 2.9 fornece uma representação visual clara e concisa do desempenho do modelo em relação a cada classe. Por ser uma matriz multiclasse, é considerada classe positiva apenas a classe verdadeira, e todas as outras classes sendo negativas.

A métrica de precisão mede a proporção de instâncias classificadas corretamente como positivas entre todas as instâncias classificadas como positivas. O *recall* avalia a capacidade do modelo em identificar todas as instâncias positivas, medindo a proporção de instâncias positivas corretamente identificadas em relação ao total de instâncias positivas. O *F1-Score* é uma métrica que combina precisão e *recall*, fornecendo uma medida única que equilibra essas duas métricas, sendo especialmente útil quando há desequilíbrio nas classes. Conhecendo o número de falsos positivos (FP), verdadeiros positivos (VP), falsos negativos (FN) e verdadeiros negativos (VN), essas métricas podem ser calculadas da seguinte forma:

$$P = \frac{VP}{VP + FP} \tag{2.2}$$

$$R = \frac{VP}{VP + FN} \tag{2.3}$$

$$F1\text{-}score = \frac{2xPxR}{P + R} \tag{2.4}$$

No contexto de problemas de classificação multiclasse, a curva ROC (*Receiver Operating Characteristic*) e a área sob a curva (AUC) (Bradley, 1997) são métricas amplamente utilizadas para avaliar a capacidade discriminativa do modelo. A curva ROC representa a taxa de verdadeiros positivos em relação à taxa de falsos positivos para diferentes valores de limiar de classificação. A AUC resume a curva ROC em um único valor numérico, fornecendo uma medida geral do poder discriminatório do modelo em relação a múltiplas classes.

Ao lidar com problemas de classificação multiclasse, a avaliação usando curva ROC e AUC pode ser realizada considerando cada classe individualmente em relação às demais classes. Uma abordagem comum é a técnica "*one-vs-all*", em que cada classe é considerada positiva enquanto as outras são agrupadas como negativas. Dessa forma, é possível calcular uma curva ROC e uma AUC para cada classe em relação às demais, permitindo uma análise mais detalhada do desempenho discriminativo do modelo em cada classe.

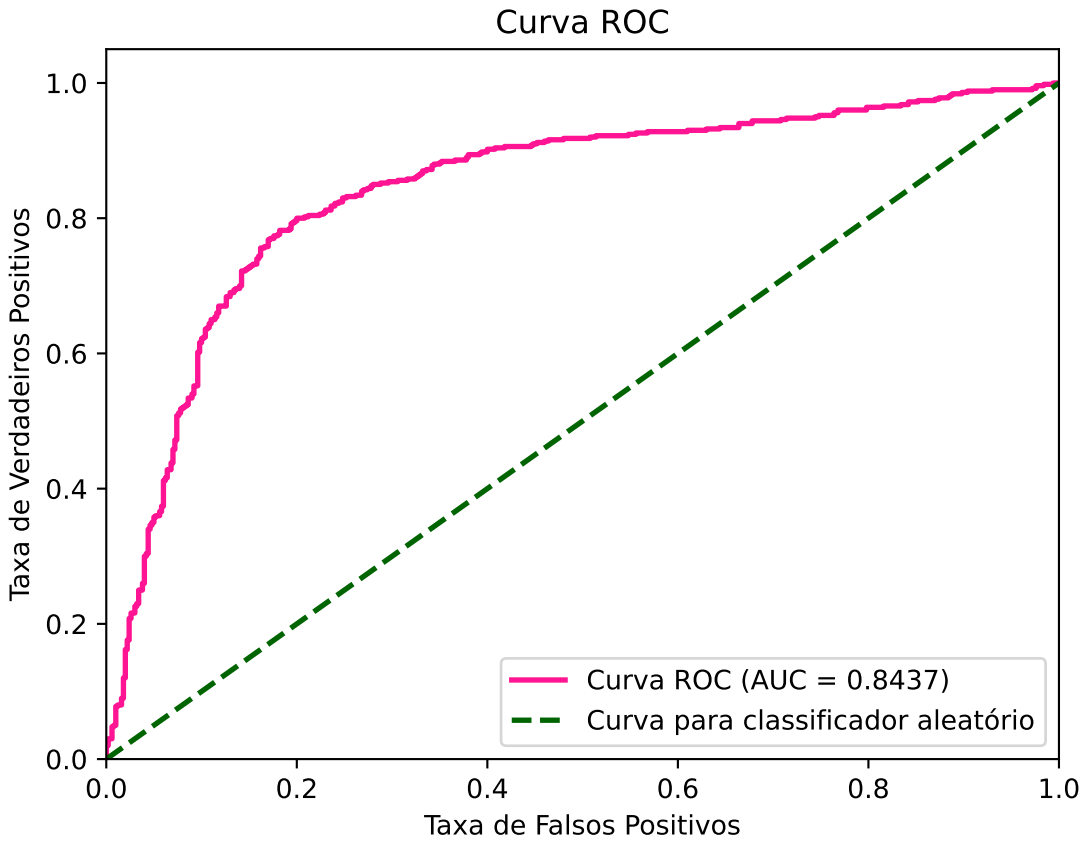

Figura 2.10 – Exemplo de curva ROC. Em rosa, uma curva ROC de um modelo com AUC = 0,8437. Em verde, a curva característica de modelo classificador aleatório.

Fonte: autoria própria.

Na Figura 2.10 é possível ver um exemplo de uma Curva ROC de um modelo em rosa. É possível ver que a AUC pode variar entre 0 e 1, e quanto maior ela é, melhor o poder de classificação do modelo. Em verde, é possível ver a curva característica de classificador aleatórios, que não possui capacidade discriminativa para acertar as classes. Neste caso, a taxa de verdadeiros positivos e a taxa de falsos positivos são iguais em todos os pontos de corte, resultado em uma curva diagonal, com AUC igual à 0,5.

2.5 *Few-shot learning*

Diferentemente dos seres humanos, que conseguem reconhecer novos objetos com base em poucas amostras, as máquinas, utilizando algoritmos de aprendizado, geralmente exigem grandes conjuntos de dados para alcançar uma precisão satisfatória na tarefa de reconhecimento. Portanto, foi desenvolvida a técnica de *few-shot learning* para lidar com situações em que uma rede neural precisa identificar objetos com base em um número reduzido de imagens (Wang *et al.*, 2020b) em casos de visão computacional. O trabalho

Dhillon *et al.* (2020) mostra uma linha de base para lidar com a tarefa de classificação de imagens utilizando essa técnica.

Essa abordagem típica envolve duas fases distintas: treinamento e teste *n-way k-shot*. Durante a fase de treinamento, o modelo é exposto a um certo conjunto de dados, e na fase de teste ele é avaliado em classes não vistas durante o treinamento, as quais possuem um número restrito de amostras. Essa técnica é de suma importância para a classificação de imagens sob limitação de dados, uma vez que o modelo não necessita ser treinado especificamente nos dados de teste. Dessa forma, um modelo previamente treinado pode ser expandido para incluir novas classes durante os testes, inclusive com o uso de apenas uma imagem por classe, caracterizando o *one-shot learning*.

Essa técnica é bastante utilizada no campo agrícola, com o trabalho Yang *et al.* (2022), que explora o uso de *few-shot learning* na agricultura inteligente, destacando métodos avançados divididos em categorias, como aumento de dados e aprendizado métrico, para superar a escassez de dados rotulados no campo da ciência das plantas, proporcionando benefícios significativos em termos de tempo e recursos financeiros.

2.5.1 Testes *n-way k-shot*

Um episódio de teste *n-way k-shot* pode ser descrito da seguinte maneira. Inicialmente, são estabelecidos dois conjuntos de dados distintos: o conjunto de suporte e o conjunto de consulta. O conjunto de suporte é composto de "k" imagens provenientes de "n" classes diversas, enquanto o conjunto de consulta consiste em "q" imagens dessas mesmas "n" classes, sem qualquer sobreposição entre os dois conjuntos. Portanto, o "n" representa a quantidade de classes que estão sendo testadas simultaneamente, o "k" representa quantas imagens de cada classe serão utilizadas como referência para a classificação, e o "q" representa quantas imagens de cada classe vão ser previstas pelo modelo a fim de medir a acurácia de teste. É possível observar que ao escolher os parâmetros, a soma entre "q" e "k" não pode ser maior que o tamanho da classe com menos dados, pois nesse caso seria impossível ter conjuntos de suporte e consulta completamente sem interseções de dados para essa classe.

Posteriormente, uma rede neural é aplicada a cada uma dessas imagens, gerando vetores de características para cada instância. Os vetores do conjunto de suporte, cujas classes são conhecidas, são então comparados com os vetores do conjunto de consulta, cujas classes são desconhecidas. Nesse contexto, a similaridade entre a imagem de consulta e o conjunto de suporte é meticulosamente avaliada, permitindo que o modelo preveja a classe de cada imagem no conjunto de consulta, podendo pertencer a qualquer uma das "n" classes.

Uma abordagem fundamental para a comparação desses vetores de características, visando alcançar acurácias superiores, é a utilização das redes prototípicas (Snell; Swersky; Zemel, 2017). Outras abordagens, como a de redes de correspondência (Vinyals *et al.*, 2016),

também são muito utilizadas nesse campo.

Dessa forma, uma fase de testes *n-way k-shot* compreende vários episódios distintos, nos quais a disposição dos conjuntos de suporte e consulta é aleatória. Ao concluir todos os episódios, a acurácia do modelo é determinada pela média aritmética das acurácias obtidas em cada um deles.

Esses testes desempenham um papel crucial, uma vez que há muitas situações em que um modelo precisa classificar apenas um número reduzido de classes simultaneamente. Mesmo quando há uma escassez de dados para essas classes, é possível alcançar altas taxas de acurácia nessa tarefa.

2.5.2 Redes Prototípicas

Introduzidas em Snell, Swersky e Zemel (2017), as redes prototípicas representam uma abordagem para lidar com desafios de *few-shot learning*. Diante da limitação significativa de dados, esses redes buscam mitigar o sobreajuste ao assumir que um classificador deve possuir um viés indutivo simples. O conceito central desas redes envolve a ideia que existe uma representação do espaço na qual os pontos se agrupam em torno de um protótipo único para cada classe. Para alcançar isso, é utilizada uma rede neural para extrair um vetor de características para cada dado, e o protótipo de uma classe é a média de todos esses vetores de uma determinada classe.

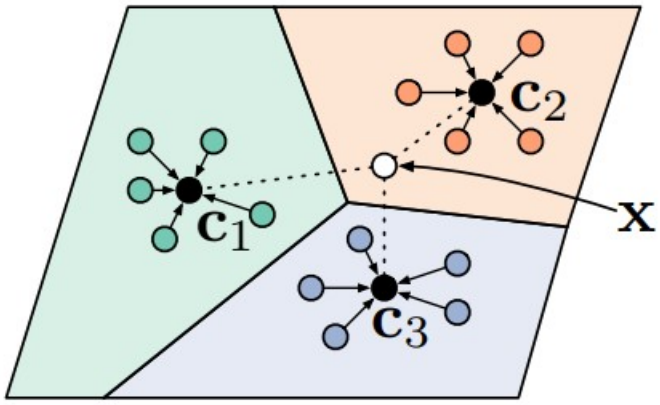

Figura 2.11 – Redes prototípicas no cenário de *few-shot learning 3-way 5-shot* em que c1, c2 e c3 representam os protótipos de cada uma das classes.

Fonte: Snell, Swersky e Zemel (2017)

Depois que todos os protótipos são calculados, as redes prototípicas empregam uma métrica de distância, como a distância euclidiana, para avaliar a proximidade entre o vetor de características do dado de consulta e os protótipos de classe. A classe atribuída ao dado de consulta é aquela cujo protótipo exibe a menor distância. Essa abordagem visa capturar a

essência das classes com base em exemplos de suporte limitados, promovendo a generalização em situações de *few-shot learning*. Em comparação com abordagens mais complexas de meta-aprendizado, as redes prototípicas se destacam por sua simplicidade e eficiência, consolidando-se como uma alternativa atrativa para enfrentar os desafios de aprendizado com poucos exemplos e, de forma geral, na busca por representações robustas e generalizáveis.

Cabe ressaltar que o uso de uma boa métrica de distância é crucial para as redes prototípicas, e algumas das principais métricas são:

a) Distância Euclidiana:

A distância euclidiana é uma medida geométrica que quantifica a separação ou a diferença entre dois pontos em um espaço euclidiano, sendo calculada como a magnitude do vetor formado pelos deslocamentos nas coordenadas correspondentes desses pontos. Dados dois pontos x e y em um espaço euclidiano de "n" dimensões, a distância euclidiana entre x e y é dada por:

$$D(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.5}$$

b) Similaridade de Cossenos:

A similaridade de cossenos é uma medida que avalia a proximidade entre dois vetores em um espaço multidimensional, medindo o cosseno do ângulo entre eles. Essa métrica também pode ser aplicada para medir a proximidade entre dois pontos representados como vetores no espaço multidimensional. Ao calcular a similaridade de cossenos entre esses vetores, considera-se tanto a direção quanto a magnitude relativa, fornecendo uma medida de proximidade entre os pontos. Essa abordagem permite avaliar a similaridade não apenas pela orientação dos vetores, mas também pela distância entre os pontos no espaço multidimensional. Essa similaridade pode ser calculada por:

$$D(x,y) = \frac{x \cdot y}{||x||\,||y||} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \tag{2.6}$$

c) Divergência de Kullback–Leibler

A divergência de Kullback-Leibler, também conhecida como entropia relativa, é uma medida que quantifica a diferença entre duas distribuições de probabilidade. Essa métrica é utilizada para avaliar a distância entre duas funções de probabilidade, indicando o quanto uma diverge da outra. Diferente das outras métricas explicadas acima, essa métrica necessita das distribuições de probabilidades. Considerando X e Y as distribuições de probabilidade para “n” diferentes classes, essa divergência pode ser calculada por:

$$D_f(x,y) = \sum_{i=1}^{n} \log \frac{X_i}{Y_i} \tag{2.7}$$

3 Materiais e Métodos

Nesta seção, será explicada a metodologia empregada para conduzir os experimentos deste trabalho. Os experimentos foram realizados utilizando as configurações de *hardware* mostradas na Tabela 3.1. Além disso, para os experimentos foram feitos códigos utilizando *Python* 3.10.11 e *PyTorch* 1.13.1.

Tabela 3.1 – Configurações de *hardware* utilizadas para os experimentos.

	Armazenamento
Memória RAM DDR4	4x 16 GB
Placa de Vídeo NVIDIA GeForce RTX 2080	2x 8 GB
SSD Seagate	1 TB
Processador Intel Core i7-8700 @ 3.20GHz	

Fonte: autoria própria.

Embora já exista um *dataset* de grande escala feito para classificação de insetos, o IP102 (Wu *et al.*, 2019), geralmente, a quantidade de fotos de determinadas espécies de insetos é muito pequena, dificultando a tarefa de criar grandes modelos de classificação para esses insetos. No próprio IP102 é possível notar uma disparidade muito grande entre a quantidade de imagens de cada classe, uma vez que a classe mais populada possui 5740 amostras, e a menos populada apenas 71. Essa desigualdade na distribuição de dados, que pode ser vista na Tabela A.10, pode afetar a eficácia dos modelos de classificação desenvolvidos com base neste conjunto de dados.

Além disso, é relevante considerar a realidade agrícola. Em uma mesma cultura de plantação em um local específico, é improvável existirem muitas espécies diferentes de insetos danosos a plantação simultaneamente. Em tal cenário, um modelo de aprendizado baseado em poucos exemplos, como o *few-shot learning*, poderia ser uma opção mais eficiente e eficaz. Diferentemente de um modelo de classificação convencional que necessita de uma grande quantidade de dados para treinamento, esses modelos podem diferenciar eficazmente pequenas classes de insetos simultaneamente com base em um número limitado de exemplos de treinamento. Assim, a adoção de tais métodos poderia melhorar a resolução e precisão do monitoramento e controle de pragas de insetos em ambientes agrícolas, especialmente para espécies que possuem poucas imagens.

Mais uma consideração adicional que merece atenção em relação ao conjunto de dados IP102 é a abordagem para diferentes estágios de vida das espécies de insetos. No IP102, a mesma espécie, independentemente do estágio da vida, é agrupada na mesma

classe. Essa metodologia, embora compreensível por sua simplicidade, pode potencialmente ignorar as diferenças significativas na aparência e comportamento entre insetos em diferentes estágios da vida, como as fases inicial e adulta. Assim, poderia ser benéfico explorar um modelo de classificação que considera separadamente esses estágios da vida. Essa mudança sutil de perspectiva pode levar a uma classificação mais matizada e, consequentemente, potencialmente melhorar a eficácia do monitoramento e controle de pragas de insetos.

Diante das limitações apresentadas pelo IP102, foi decidido a utilização do *dataset* IP-FSL, apresentado em Gomes e Borges (2022). Este *dataset* é um aprimoramento do IP102 para utilização de *few-shot learning*, em que a partir do IP102, foram criadas 97 classes de insetos em estágio adulto e 45 classes de insetos em estágio inicial de vida. Para ser considerado estágio inicial, foram consideradas imagens com a presença de ovo, larva ou pupa. Como o IP102 tem grande desbalanceamento das classes, no IP-FSL foi decidido limitar todas as classes a um máximo de 50 imagens, o que resultou num total de 2050 imagens das 45 classes de estágio inicial e 4767 imagens das 97 classes de fase adulta, totalizando 6817 imagens de insetos. A distribuição de imagens por classe pode ser vista na B.11, e é possível observar que todas as 102 classes do IP102 foram utilizadas neste conjunto de dados.

Além disso, a *Swin Transformer* com suas inovações se mostra como uma boa alternativa para classificação de imagens, então foram feitos experimentos com essa arquitetura utilizando o IP-FSL. No artigo de referência deste *dataset* (Gomes; Borges, 2022), também foram feitos experimentos com *few-shot learning* para a classificação de insetos, utilizando um extrator de características próprio baseado em RNCs, etapa de treinamento com meta-aprendizado e testes *n-way k-shot*. Ao fazer testes com os mesmo parâmetros e mesmo conjunto de dados que este trabalho, é possível estabelecer uma comparação entre os resultados e o desempenho da *Swin Transformer* nesta tarefa.

Portanto, diante de tudo isso, neste trabalho foram realizados 5 experimentos:

a) Experimento I: consiste em testar diferentes versões da *Swin Transformer* e verificar qual é a melhor para utilizar nos experimentos posteriores.

b) Experimentos II e III: consistem em testar *few-shot learning* nas classes de estágio adulto do IP-FSL, utilizando o melhor modelo obtido no Experimento I.

c) Experimentos IV e V: consistem em testar *few-shot learning* nas classes de estágio inicial do IP-FSL, utilizando o melhor modelo obtido no Experimento I.

As diferenças entre os experimentos serão melhor explicadas nas seções a seguir.Na Figura 3.12 é possível observar o fluxo de execução dos experimentos deste trabalho.

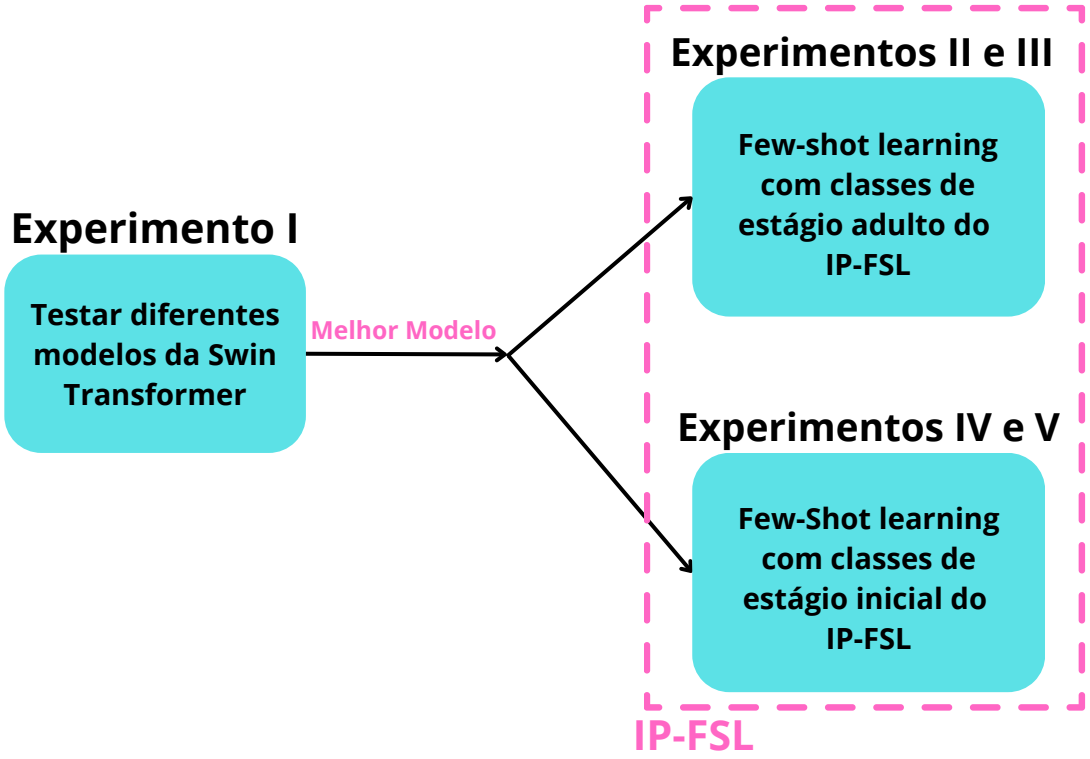

Figura 3.12 – Fluxo de execução dos experimentos do trabalho.

Fonte: autoria própria.

3.1 Experimento I: diferentes modelos da *Swin Transformer*

O Experimento I consiste em verificar a acurácia de diferentes versões da *Swin Transformer* como extratora de características em tarefas de classificação de insetos. A *Swin Transformer* surge como uma abordagem inovadora neste âmbito amplamente dominado por estratégias que envolvem redes neurais convolucionais. Como conjunto de dados para este experimento, foi utilizado o *dataset* IP102 (Wu *et al.*, 2019) e todas as suas 102 classes. Este *dataset* foi escolhido devido a sua grande variedade de classes, além do grande número de quantidade de imagens. Como esse experimento visa apenas verificar a capacidade de extrair características de imagens de insetos da *Swin Transformer*, e não fazer testes com *few-shot learning*, o IP102 foi preterido em relação ao IP-FSL. Neste experimento foram utilizadas apenas 10% das imagens do IP102, a fim de se obter resultados mais rápidos. Mesmo com apenas uma parte do *dataset*, por serem experimentos comparativos, é possível conseguir os resultados necessários de forma mais rápida utilizando apenas estes 10%. Essa divisão do *dataset* foi feita de forma aleatória, mantendo a proporção de imagens em cada classe, ou seja, foram utilizadas 10% das imagens de cada uma das classes escolhidas aleatoriamente.

Em relação à separação dos dados para esses experimentos, utilizou-se a técnica de validação cruzada *k-fold* e apenas conjuntos de treinamento e teste foram utilizados. Portanto, todas as imagens correspondentes à 10% do IP102 foram agrupadas, e a separação das imagens para cada *fold* foi realizada de forma aleatória.

Para a realização do experimento, foram utilizados modelos pré-treinados pelos próprios criadores da *Swin Transformer*, disponibilizados no *Github* oficial do modelo (Microsoft, 2021). Foram escolhidos os três modelos com maiores dimensões listados a seguir:

a) *swin large patch4 window12 384 22kto1k* (*Swin-L FT*)

   O modelo é uma variante do *Swin Large*, que foi pré-treinado no conjunto de dados *ImageNet 22k*. Durante o pré-treinamento, foram utilizados *patches* de 4 *pixels*, janelas de 12 *pixels* e imagens de entrada com resolução de 384×384 *pixels*. Após essa fase inicial de pré-treinamento, o modelo foi submetido a um processo de *fine-tuning* usando o conjunto de dados *ImageNet 1k*.

b) *swin large patch4 window12 384 in22k* (*Swin-L*)

   Esse modelo é uma versão da *Swin Large*, que passou por um pré-treinamento no conjunto de dados *ImageNet 22k* usando *patches* de 4 *pixels* e janelas de 12 *pixels* em imagens de 384×384 *pixels*.

c) *swinv2 large window12to24 192to384 22kft1k* (*SwinV2-L*)

   Esse modelo é uma versão do *SwinV2 Large* que passou por um pré-treinamento no conjunto de dados *ImageNet 22k* usando *patches* de 4 *pixels* e janelas de 12 *pixels* em imagens de 192×192 *pixels*. Em seguida, foi submetido a um *fine-tuning* com o *ImageNet 1k*, utilizando janelas de 24 *pixels* e imagens de entrada de resolução 384×384 *pixels*.

Esses modelos pré-treinados então foram submetidos a um processo de *fine-tuning*, agora no conjunto de dados do experimento, que corresponde a 10% do IP102. Todos os modelos foram avaliados sob as mesmas condições de *hardware* e aleatoriedade das imagens. Para o treinamento destes modelos, foi utilizado o método de validação cruzada *10-fold* com os parâmetros da Tabela 3.2.

Para este treinamento, inicialmente, a última camada do modelo foi substituída por uma camada linear, e os gradientes de todas as camadas originais foram desativados, mantendo apenas os gradientes da última camada ativos. Em seguida, o treinamento foi iniciado, consistindo em várias épocas, onde cada época envolveu uma passagem completa pelos conjuntos de treinamento e teste. Durante essas épocas, a otimização dos pesos visou minimizar uma função de perda, que quantifica a diferença entre as previsões do modelo e os rótulos verdadeiros. O otimizador, utilizando os gradientes dessas diferenças, ajustou gradualmente os pesos da última camada, otimizando assim o desempenho do modelo. Após cada época, o modelo foi avaliado no conjunto de teste sem modificar nenhum peso, apenas

para verificar seu desempenho. Esse processo de treinamento ocorreu 10 vezes, utilizando um *fold* diferente como conjunto de teste em cada iteração, enquanto os outros 9 serviram como conjunto de treinamento. O melhor modelo obtido é apresentado no Capítulo 4.

Tabela 3.2 – Parâmetros utilizado no treinamento dos modelos.

Parâmetro	
Função de Perda	Entropia Cruzada
Otimizador	Adam
Taxa de Aprendizado	0,001
Número de Épocas	10
Número de *Folds*	10
Tamanho do *batch*	16 e 64

Fonte: autoria própria.

3.2 Experimento II: *few-shot learning* para classes adultas com divisão por classes

O Experimento II se concentra exclusivamente no uso do conjunto de dados das classes adultas do IP-FSL, conforme definido em Gomes e Borges (2022), para a realização de *few-shot learning* para classificação de insetos. O modelo com melhor desempenho no Experimento I foi escolhido para esta etapa. O conjunto de dados foi dividido, alocando 80% das classes no conjunto de treino e as restantes 20% no conjunto de testes. Essa abordagem foi adotada para assegurar que nenhuma das classes testadas fosse previamente vista no processo de treinamento, permitindo avaliar a capacidade do modelo de aprender a classificar novas classes com apenas uma ou cinco amostras.

Durante o treinamento, foi aplicado um *fine-tuning* ao modelo por meio de aprendizado supervisionado convencional, seguindo a metodologia do Experimento I e utilizando os parâmetros apresentados na Tabela 3.2. A opção por essa estratégia, em vez de treinar exclusivamente com estratégias de meta-aprendizado, visa garantir a obtenção do melhor extrator de características possível. Isso, por sua vez, contribui para alcançar melhores resultados de acurácia na fase de teste com *few-shot learning*. Durante o treinamento, o modelo se especializa em extrair características de fotos de insetos na fase adulta, assegurando uma elevada acurácia na fase de teste, mesmo para classes diferentes.

É importante ressaltar que o modelo utilizado, o *Swin Transformer*, foi pré-treinado no *ImageNet* (Deng *et al.*, 2009), ou seja, já tinha bons pesos ajustados para reconhecimento de objetos. Durante o *fine-tuning* do modelo com o conjunto de insetos, os pesos foram

levemente ajustados para aprimorar a tarefa de classificação de insetos na fase adulta. Esse ajuste leve nos pesos durante o *fine-tuning* contribui para garantir que a rede neural ainda consiga generalizar para imagens de novas classes durante a etapa de teste.

Tabela 3.3 – Métricas para a avaliação do modelo de *few-shot learning*.

Métricas
Similaridade de cossenos
Divergência de Kullback-Leibler

Fonte: autoria própria.

Então, na fase de teste, foram feitos testes com *few-shot learning*. Para este propósito, foi utilizada uma abordagem com redes prototípicas, empregando o modelo *Swin Transformer* pré-treinado e com *fine-tuning* sobre os dados de treinamento como extrator de características das imagens. A preferência por redes prototípicas em vez de redes de correspondência foi sustentada pelo estudo de Gomes e Borges (2022), que revelou maior precisão no conjunto de dados *Mini-ImageNet* ao lidar com tarefas de *few-shot learning*. Adicionalmente, as métricas referenciadas na Tabela 3.3, foram utilizadas para medir as distâncias entre pontos usando a rede prototípica, permitindo a comparação dos resultados.

A fase de teste foi realizada por meio de diversos episódios, sendo que cada episódio consistia em uma seleção aleatória de "k" imagens provenientes de "n" classes distintas para compor o conjunto de suporte. A *Swin Transformer* foi aplicada individualmente a cada imagem, resultando na geração de um vetor de características correspondente. Simultaneamente, também de maneira aleatória, "q" imagens de cada uma das "n" classes foram escolhidas, constituindo, assim, o conjunto de consulta. Para este conjunto, vetores de características também foram gerados.

Tendo conhecimento das classes associadas aos vetores de características do conjunto de suporte, foi calculado o protótipo de cada uma dessas classes, e a diferença entre cada imagem do conjunto de consulta e cada protótipo do conjunto de suporte foi medida utilizando métricas, como aquelas da Tabela 3.3. Com base nessas diferenças, foi possível fazer previsões das classes de cada imagem do conjunto de consulta. Essas previsões foram realizadas em todos os episódios, permitindo a medição da acurácia global do modelo.

No intuito de explorar a sensibilidade do modelo a diferentes configurações, realizaram-se testes variando os valores de *n-way*, *k-shot* e *query*. A escolha de diferentes combinações desses parâmetros permitiu uma análise abrangente do desempenho do modelo em condições diversas de *few-shot learning*. Vale ressaltar que, durante a etapa de teste, a distinção entre as imagens de suporte e as imagens de consulta foi crucial para avaliar a capacidade

do modelo em generalizar o aprendizado de poucos exemplos. Foram feitos experimentos *three-way* e *five-way*, com *one-shot* e *five-shot*, mantendo o valor de "q" que é o valor de imagens de consulta, fixado em 5.

3.3 Experimento III: *few-shot learning* para classes adultas com divisão estratificada

O Experimento III, semelhante ao experimento anterior, concentra-se exclusivamente nas classes adultas do IP-FSL, utilizando o modelo com melhor desempenho do Experimento I. A principal distinção em relação ao Experimento II reside na abordagem de divisão do conjunto de dados. Enquanto no experimento anterior o *dataset* foi dividido por classes, neste experimento adotou-se uma abordagem seletiva, considerando a quantidade de imagens disponíveis por classe, mas garantindo a presença de todas as classes em ambos os conjuntos do experimento.

Para as classes com 50 imagens, foi utilizada a estratégia de utilizar 80% das imagens para treino, e 20% das imagens para teste. No entanto, para as classes com menos de 50 imagens, foi necessário garantir um mínimo de 10 imagens no conjunto de teste para permitir a realização de testes com *five-shot* e $q = 5$. Portanto, para essas classes, foram escolhidas 10 imagens aleatórias para o conjunto de teste e as imagens restantes dessas classes foram alocadas para o conjunto de treinamento. Essa escolha de fixar 10 imagens de teste para todas as classes foi feita, pois apenas uma pequena quantidade de classes possui menos de 50 imagens, e essas poucas imagens que foram "perdidas" pelo conjunto de treino pouco afetariam o treinamento do modelo, mas permitiriam fazer testes com *five-shot* e $q = 5$, assim como é feito por convenção em diversos trabalhos de *few-shot learning*.

Neste experimento o treinamento foi realizado da mesma forma que no experimento anterior, e também foram feitos testes *n-way k-shot* com "n" variando entre 3 e 5, "k" variando entre 1 e 5 e $q = 5$.

3.4 Experimento IV: *few-shot learning* para classes iniciais com divisão por classes

O Experimento IV foi feito seguindo um protocolo metodológico semelhante ao Experimento II, adotando a mesma estrutura experimental e parâmetros, com a exceção do conjunto de dados utilizado. O Experimento IV utilizou o conjunto de dados com os insetos em fase inicial da vida do IP-FSL, diferente do experimento anterior que usou as imagens de insetos na fase adulta. O modelo selecionado para este experimento foi o mesmo que apresentou o melhor desempenho no Experimento I. A divisão do novo conjunto de dados

foi realizada da mesma maneira que a do Experimento II, separando 80% das classes para o conjunto de treino, e 20% para conjunto de testes, garantindo que as classes dos testes fossem totalmente novas para o modelo. Durante a fase de treinamento, o modelo passou por *fine-tuning* com aprendizado supervisionado, seguindo os mesmos parâmetros estabelecidos anteriormente.

Na etapa de teste, o experimento IV também adotou a abordagem de *few-shot learning*, utilizando redes prototípicas, e empregou as mesmas métricas para avaliação. A análise de sensibilidade do modelo explorou variações nos parâmetros n-way, k-shot e query, enquanto a diferenciação entre imagens de suporte e consulta permaneceu fundamental para avaliar a capacidade do modelo em generalizar a partir de exemplos limitados. Foram feitos experimentos *three-way* e *five-way*, com *one-shot* e *five-shot*, com $q = 5$, assim como nos outros experimentos. Essa consistência metodológica permite uma comparação direta entre os resultados dos Experimentos II e IV, destacando as nuances resultantes das diferenças nos conjuntos de dados.

3.5 Experimento V: *few-shot learning* para classes iniciais com divisão estratificada

Assim como o Experimento IV foi feito de forma análoga ao Experimento III, este experimento V também foi feito de maneira análoga ao Experimento III. Foram utilizadas as imagens de insetos em fase inicial de vida, juntamente com o modelo que melhor performou no experimento I. O *dataset* foi dividido de forma seletiva, garantindo 10 imagens de teste para todas as classes, e com o resto das imagens no conjunto de treinamento. Com essa divisão, todas as classes estavam presentes tanto no conjunto de testes, quanto no conjunto de treino. Assim como no experimento III, foi feito um *fine-tuning* do modelo *Swin Transformer* com os dados de treinamento, e esse modelo resultante foi utilizado como extrator de características para os testes *n-way k-shot* feitos, com "n" variando entre 3 e 5, "k" variando entre 1 e 5, e q = 5. Mantendo a consistência metodológica como foi feito, é possível fazer uma comparação direta dos resultados desse experimento, tanto com o Experimento III quanto com o Experimento IV.

4 Resultados e Discussões

4.1 Experimento I: diferentes modelos da *Swin Transformer*

Após a realização do experimento, os resultados foram mostrados na Tabela 4.4. Para o cálculo da acurácia de cada modelo, foi feita a média aritmética das acurácias dos 10 *folds*, e para o cálculo da AUC, foi feita uma média ponderada da AUC de cada uma das 102 classes. O modelo SwinV2-L FT não pôde ser testado com um *batch* de 64 imagens devido a limitações de *hardware*, por ser um modelo com mais parâmetros que os outros dois. Nas Figuras 4.13, 4.14, 4.15, 4.16, 4.17, podem ser vistas as curvas ROC de cada um dos modelos.

Tabela 4.4 – Resultado obtidos após o treinamento dos modelos.

Modelo	Tamanho do *batch*	Acurácia	AUC	Tempo de Treinamento
Swin-L FT	16	0,6573	0,9701	11h08m51s
Swin-L FT	64	0,6615	0,9740	7h46m40s
Swin-L	16	0,6629	0,9717	12h02m40s
Swin-L	64	**0,6673**	**0,9747**	**7h30m46s**
SwinV2-L FT	16	0,6458	0,9591	14h21m53s

Fonte: autoria própria.

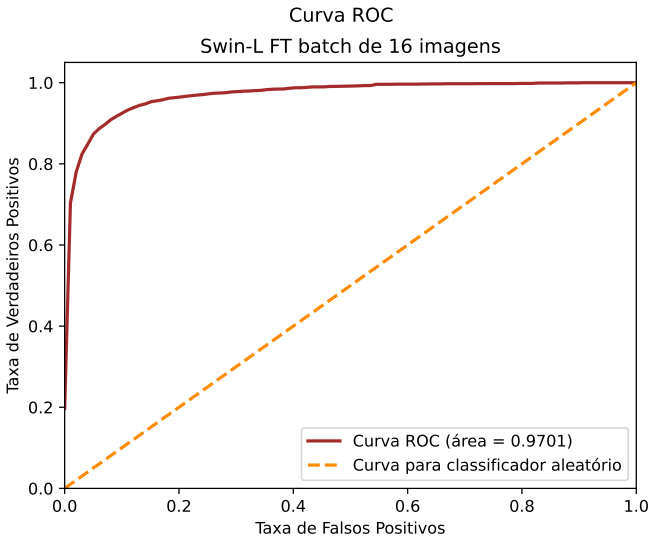

Figura 4.13 – Curva ROC obtida pelo modelo Swin-L FT com *batch* de 16 imagens e sua respectiva AUC.

Fonte: autoria própria.

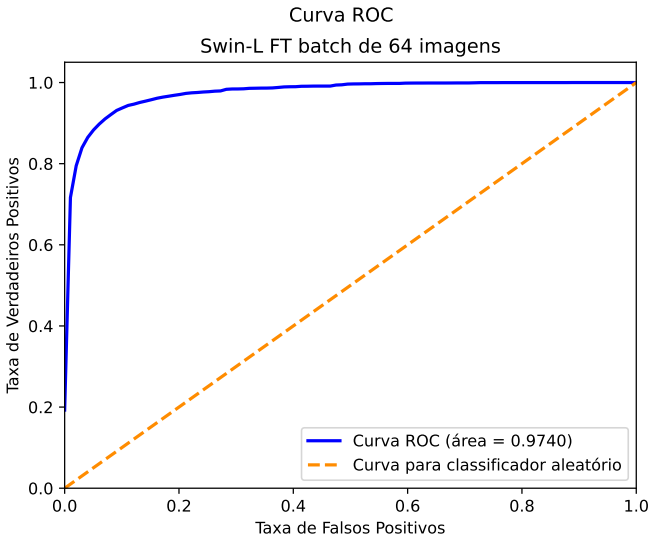

Figura 4.14 – Curva ROC obtida pelo modelo Swin-L FT com *batch* de 64 imagens e sua respectiva AUC.

Fonte: autoria própria.

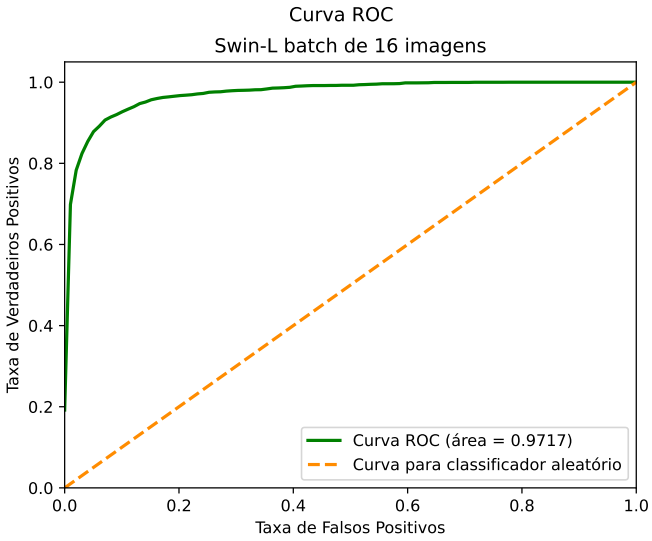

Figura 4.15 – Curva ROC obtida pelo modelo Swin-L com *batch* de 16 imagens e sua respectiva AUC.

Fonte: autoria própria.

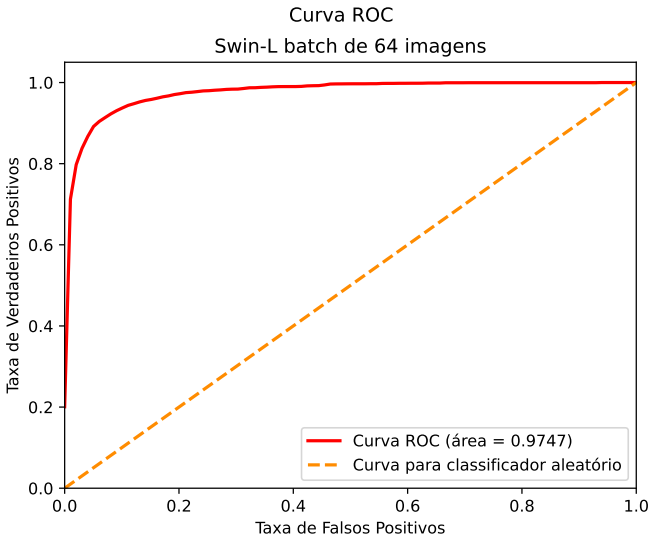

Figura 4.16 – Curva ROC obtida pelo modelo Swin-L com *batch* de 64 imagens e sua respectiva AUC.

Fonte: autoria própria.

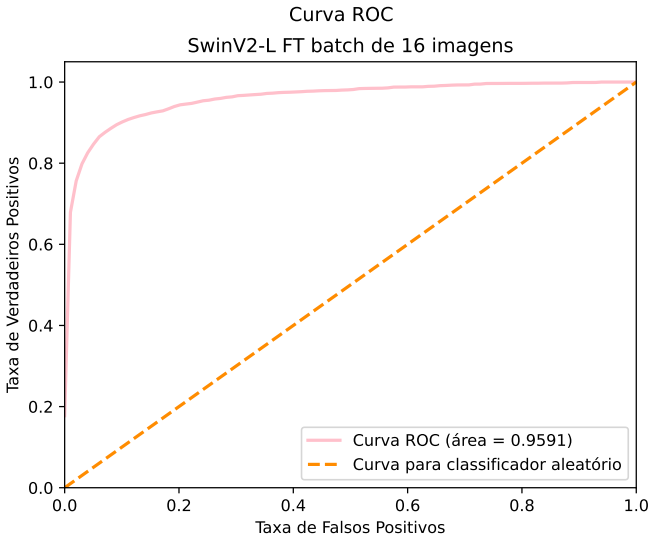

Figura 4.17 – Curva ROC obtida pelo modelo SwinV2-L FT com *batch* de 16 imagens e sua respectiva AUC.

Fonte: autoria própria.

Observando os resultados da Tabela 4.4, é possível perceber que o modelo Swin-L com um *batch* de 64 imagens foi o que obteve melhores resultados, tanto de acurácia, quanto de AUC, além de ter sido o modelo com o menor tempo de treinamento. Foi possível observar um desempenho inferior do modelo SwinV2-L FT com relação aos outros modelos, o que pode parecer estranho, porém era esperado, uma vez que este modelo, apesar de possuir mais parâmetros, foi pré-treinado com imagens de 192×192 *pixels*, o que causou uma grande

perda de informações durante este pré-treino. Este era o modelo com mais parâmetros e pré-treinado disponibilizado pelos autores da SwinV2, e por isso foi o escolhido.

É notável que, mesmo utilizando apenas 10% do conjunto de dados IP102, todos os modelos alcançaram resultados de acurácia superiores aos obtidos no artigo de referência do dataset (Wu *et al.*, 2019). O estudo original, deste artigo referência. que empregou a ResNet, atingiu uma acurácia máxima de 49,4% na classificação de insetos. Essa comparação evidencia o desempenho destacado da *Swin Transformer* em relação a uma rede neural convolucional na mesma tarefa.

Portanto, pelo modelo Swin-L com *batch* de 64 imagens ter se mostrado o melhor, na Tabela 4.5 podem ser vistos os resultados de acurácia para cada um dos *folds* de treinamento do modelo Swin-L com *batch* de 64 imagens.

Tabela 4.5 – Resultados de acurácia obtidos em cada um dos 10 *folds* da validação cruzada pelo modelo SwinL com *batch* de 64 imagens.

Número do Fold	Resultado de acurácia por *fold*
1	0,6739
2	0,6631
3	0,6901
4	0,6626
5	0,6274
6	0,6707
7	0,6748
8	0,6721
9	0,6626
10	0,6762

Fonte: autoria própria.

Além disso, na Tabela C.12 podem ser vistas todas as métricas de cada uma das 102 classes do IP102 utilizadas no experimento, no treinamento do melhor modelo. Nesta tabela é possível ver não só o desbalanceamento das classes, mas também o quanto a falta de imagens afeta nos resultados das classes minoritárias. É possível notar que as classes com mais imagens, como a dos insetos "*Lycorma delicatula*", "*Miridae*" e "*Cicadellidae*", que são as únicas que possuem mais de 500 imagens testadas, obtiveram *f1-score* de 0,84, 0,85 e 0,86, respectivamente, enquanto as classes com até 10 imagens obtiveram um *f1-score* médio de apenas 0,25. Mesmo se tratando apenas de 10% do IP102, que foi o conjunto de dados utilizado para este experimento, essa divisão foi feita de maneira estratificada, ou seja, a proporção das classes foi mantida, então mesmo com o *dataset* completo esse

desbalanceamento das classes ainda existe e afeta os resultados das classes minoritárias.

Portanto, diante desse desbalanceamento do IP102, se justifica a utilização do IP-FSL nos experimentos posteriores, vinculado a testes com *few-shot learning*, para conseguir realmente fazer testes considerando poucas imagens para todas as classes, que é o mais comum na realidade agrícola. Além disso, como o modelo Swin-L com *batch* de 64 imagens foi o que melhor performou nesse experimento, ele foi utilizado nos Experimentos II, III, IV e V como extrator de características das imagens.

4.2 Experimento II: *few-shot learning* para classes adultas com divisão por classes

O Experimento II foi realizado e os resultados foram expostos na Tabela 4.6. Nesta tabela é possível observar ótimos resultados nos testes feitos. Em Gomes e Borges (2022), foi realizado um experimento semelhante, com o mesmo *dataset*, mesma divisão dos dados e mesma quantidade de episódios de teste, porém com duas grandes diferenças: a primeira, é que neste trabalho foi utilizado um extrator de características próprio baseado em RNCs ao invés da *Swin Transformer*. A segunda diferença é que na etapa de treinamento, foram feitas episódios de treinamento com meta-aprendizado, diferente deste trabalho que foi focado em fazer um *fine-tuning* da rede neural com aprendizado supervisionado durante a etapa de treinamento.

Tabela 4.6 – Resultados obtidos no Experimento II, que envolve o conjunto de dados de insetos adultos com divisão do *dataset* por classes. SC: Similaridade de Cossenos, KL: Divergência de Kullback-Leibler

N-way	One-shot		Five-shot	
	SC	KL	SC	KL
Three-way	0,8841	**0,8937**	**0,9605**	0,9377
Five-way	0,8162	**0,8238**	**0,9252**	0,8940

Fonte: autoria própria.

Com essa escolha aliada a *Swin Transformer*, foi possível obter resultados de acurácia impressionantes de até 96,05%, mesmo para uma classe com apenas 5 imagens que não foi vista durante o treinamento, e resultados de 89,37% para uma classe com apenas uma imagem de referência. O Trabalho (Gomes; Borges, 2022) obteve resultados de acurácia de 77,97% em *three-way one-shot*, 66,94% em *five-way one-shot*, 86,33% em *three-way five-shot* e 77,68% em *five-way five-shot*, ou seja, a utilização da *Swin Transformer* aliada ao seu *fine-tuning* conseguiu atingir resultados melhores em todos os testes desse experimento.

4.3 Experimento III: *few-shot learning* para classes adultas com divisão estratificada

O Experimento III foi realizado e os resultados foram mostrados na Tabela 4.7. Após a etapa de treinamento, apenas para ter uma dimensão do desempenho do modelo, foi feito um teste no conjunto de testes com as 97 classes e foi obtido um resultado de 66,08% de acurácia na previsão das classes.

Tabela 4.7 – Resultados obtidos no Experimento III, que envolve o conjunto de dados de insetos adultos com divisão estratificada do *dataset*. SC: Similaridade de Cossenos, KL: Divergência de Kullback-Leibler

N-way	One-shot		Five-shot	
	SC	KL	SC	KL
Three-way	0,8965	**0,9370**	0,9576	**0,9703**
Five-way	0,8351	**0,8991**	0,9312	**0,9414**

Fonte: autoria própria.

A principal diferença entre este experimento e o Experimento II é a divisão do *dataset*, em que neste experimento todas as classes estiveram presentes em ambos os processos de treinamento e teste. Por conta disso, como as classes de teste já haviam sido vistas no processo de treinamento, era esperado obter resultados um pouco melhores do que os do Experimento II, conforme pode ser observado. Ainda sim, esse experimento se trata de um aprendizado com poucas amostras, pois com apenas 50 amostras de cada classe, foi possível obter uma acurácia de 97,03% em testes com 3 classes diferentes. Esses resultados mostram que mesmo com essa quantidade pequena de amostras, é possível obter resultados excelentes em classificação de insetos adultos.

4.4 Experimento IV: *few-shot learning* para classes iniciais com divisão por classes

Na Tabela 4.8, são mostrados os resultados do Experimento IV. Observando os resultados, é possível observar que houve uma piora do desempenho do modelo com relação às imagens do estágio adulto. Essa queda pode ser atribuída à notável semelhança entre as imagens das classes, que frequentemente apresentam características muito próximas em diversos estágios. A Figura 4.18 ilustra a marcante semelhança entre várias espécies nesse estágio de vida, justificando, em parte, um desempenho pouco inferior. Para efeito de comparação, a Figura 4.19 mostra imagens dessas mesmas espécies, porém na fase adulta, sendo possível observar que nessa etapa da vida as espécies são mais diferenciáveis.

Tabela 4.8 – Resultados obtidos no Experimento IV, que envolve o conjunto de dados de insetos em estágio inicial de vida com divisão do *dataset* por classes. SC: Similaridade de Cossenos, KL: Divergência de Kullback-Leibler

N-way	One-shot		Five-shot	
	SC	KL	SC	KL
Three-way	**0,7564**	0,7246	**0,8796**	0,8438
Five-way	**0,6346**	0,5996	**0,7952**	0,7507

Fonte: autoria própria.

Mesmo diante dessas dificuldades, o modelo conseguiu alcançar uma acurácia de 87,96% para classes que nunca foram vistas durante a etapa de treinamento, utilizando apenas 5 imagens dessas classes. Além disso, obteve uma acurácia de 75,64% para classes utilizando apenas uma imagem. Esses resultados são comparáveis aos do estudo de Gomes e Borges (2022), indicando que a *Swin Transformer* não se destaca tanto como extratora de características de insetos na fase inicial da vida quanto na fase adulta.

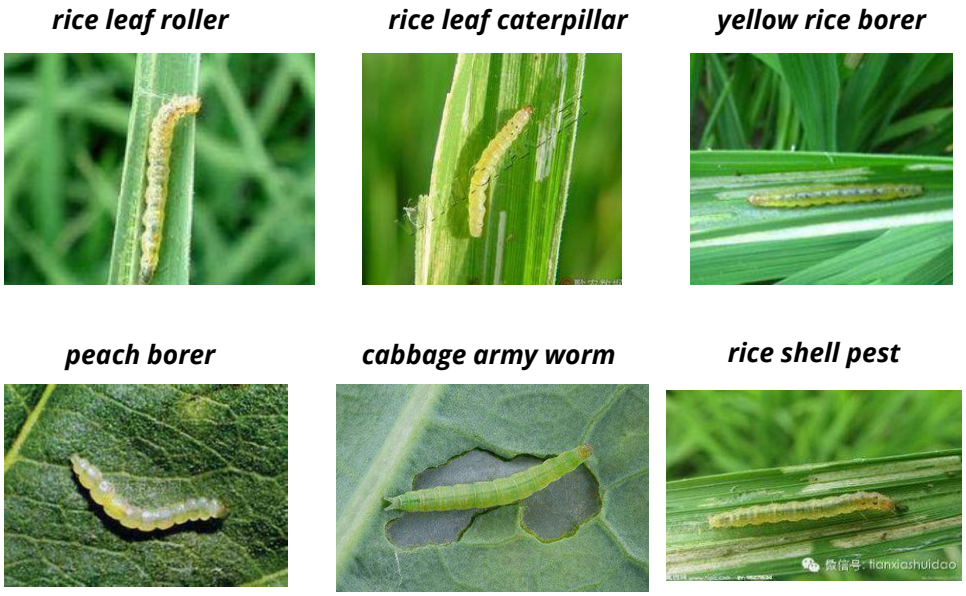

Figura 4.18 – Comparação entre imagens de seis diferentes classes do *dataset* de insetos em estágio inicial de vida, mostrando a grande semelhança entre muitas espécies nessa etapa de vida. Foram utilizados os nomes das espécies disponibilizados pelos criadores do IP102.

Fonte: autoria própria.

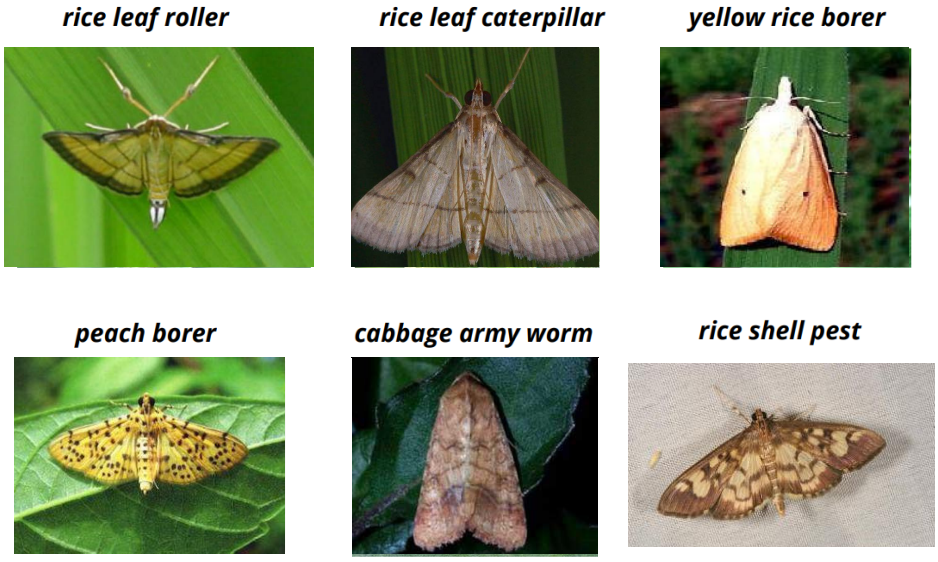

Figura 4.19 – Comparação entre imagens de seis diferentes classes do *dataset* de insetos de estágio adulto, mostrando que nesta etapa da vida os insetos são mais diferenciáveis se comparados à etapa inicial de vida.

Fonte: autoria própria.

4.5 Experimento V: *few-shot learning* para classes iniciais com divisão estratificada

Os resultados do Experimento V estão apresentados na Tabela 4.9. Durante o treinamento, foi realizado um teste abrangente com todo o conjunto de dados de teste para avaliar o desempenho do modelo. A acurácia obtida foi de 60,22%, sendo inferior à acurácia alcançada nas classes adultas durante o mesmo experimento, a qual foi de 66,08%. Isso ressalta mais uma vez a capacidade da *Swin Transformer* em distinguir as classes adultas com maior facilidade.

É relevante notar que esse desafio persiste mesmo diante da diminuição significativa no número de classes no conjunto de dados de estágio inicial, que é de 45 neste *dataset* em oposição às 97 classes de estágio adulto. .

Tabela 4.9 – Resultados obtidos no Experimento V, que envolve o conjunto de dados de insetos em estágio inicial de vida com divisão estratificada do *dataset*. SC: Similaridade de Cossenos, KL: Divergência de Kullback-Leibler

N-way	One-shot		Five-shot	
	SC	KL	SC	KL
Three-way	0,7920	**0,8431**	0,8889	**0,8991**
Five-way	0,6922	**0,7667**	0,8232	**0,8498**

Fonte: autoria própria.

Observando os resultados da tabela, é possível observar que houve uma melhora nos resultados com relação ao Experimento IV, o que era esperado, uma vez que neste experimento todas as classes utilizadas no conjunto de testes também foram utilizadas no conjunto de treino. Também foi possível perceber uma queda de desempenho do modelo se comparado ao Experimento III, em que foi feito um experimento igual, porém com o *dataset* de estágio adulto, mostrando novamente a maior dificuldade que a *Swin Transformer* apresenta em diferenciar esses insetos em estágio inicial de vida. Porém, mesmo com essas dificuldades, foi possível conseguir um desempenho de quase 90% de acurácia em identificar esses insetos, em comparações com 3 classes juntas, o que é um ótimo resultado considerando apenas 50 imagens no máximo de cada uma das classes.

4.6 Limitações

Todo experimento possui suas limitações, e os realizados neste trabalho não são exceções. Inicialmente, com o IP-FSL, só foi viável classificar as classes do IP102, que abrangem exclusivamente insetos prejudiciais a oito culturas específicas: milho, arroz, trigo, beterraba, alfafa, uva, frutas cítricas e manga. Essa limitação, no entanto, não diminui a relevância dos resultados obtidos, sugerindo que desempenhos similares podem ser alcançados em outras culturas, mesmo com um número reduzido de imagens disponíveis.

Cabe ressaltar que este estudo concentrou-se exclusivamente na classificação de insetos. Dado o êxito da *Swin Transformer* como uma eficaz extratora de características para imagens de insetos, seria pertinente explorar também uma abordagem para a detecção de insetos. Essa abordagem envolveria identificar a posição exata dos insetos nas imagens, ampliando as possibilidades de aplicação e enriquecendo a análise proposta neste trabalho.

Ademais, é importante salientar que, embora os resultados tenham sido promissores, estudos futuros poderiam expandir a avaliação para incluir diversas culturas de plantação, a fim de validar a generalização do modelo proposto. A diversidade de culturas pode apresentar desafios adicionais, mas a robustez da *Swin Transformer* sugere que ela pode ser adaptada com sucesso para diferentes contextos, mesmo quando a quantidade de imagens disponíveis é limitada.

5 Conclusões

Neste trabalho foram implementados métodos para abordar o desafio de classificação de insetos danosos a plantações no ambiente agrícola, considerando a escassez de imagens de diversas espécies. A eficácia da arquitetura *Swin* nesta tarefa foi investigada inicialmente, destacando-se pela estratégia de empregar janelas deslocadas para o processamento eficiente das imagens e alcançar uma acurácia notável na categorização desses insetos, revelando-se como uma extratora de características eficiente.

Foi possível observar que ao utilizar um modelo pré-treinado no maior conjunto de imagens possível, com a maior resolução possível, como foi o *Swin Large* que foi pré-treinado no *dataset ImageNet* 22k, que contém aproximadamente 22 mil diferentes classes em imagens de 384×384 *pixels*, foi possível obter melhores resultados. A utilização de um *batch* de 64 imagens durante o treinamento não apenas demonstrou ser mais eficiente em termos de velocidade, mas também resultou em acurácias aprimoradas em comparação com *batches* de 16 imagens.

Comparando o desempenho da *Swin Transformer* com a ResNet utilizada em Wu *et al.* (2019), foi possível observar uma grande melhora de 49,4% para 66,73% de acurácia, mesmo usando uma parte menor do conjunto de dados no caso do experimento com a *Swin*, mostrando que o uso de *transformers* em classificação de insetos pode ser uma ótima alternativa ao uso de redes neurais convolucionais.

Também foi possível observar que o desbalanceamento das classes do IP102, que é o primeiro e único *dataset* de larga escala com insetos até o momento, interfere muito nos resultados das classes minoritárias, e que optar por estratégias para lidar com a escassez de imagens no campo agrícola talvez seja a melhor alternativa, como no caso do *few-shot learning*.

Tendo investigado o funcionamento da *Swin Transformer* como extratora de características de imagens, foi possível realizar testes utilizando o IP-FSL. A escolha do IP-FSL revelou-se crucial, destacando sua importância devido à divisão entre classes adultas e classes em estágio inicial de vida. Esta distinção é essencial, uma vez que as espécies apresentam diferenças marcantes em diferentes fases de desenvolvimento. Além disso, o IP-FSL caracteriza-se por uma quantidade limitada de imagens por classe, refletindo de maneira fiel à realidade na área de estudo.

Realizando testes com *few-shot learning* nas classes adultas, foi possível observar um ótimo desempenho do modelo na classificação de insetos, mesmo em classes com pouquíssimas imagens de referência, classes essas que não foram utilizadas durante o treinamento. Para essas classes, foi possível obter resultados de 89,37%, mesmo com apenas uma imagem

de referência, conseguindo se estender até 96,05% com o número de cinco imagens. A utilização da *Swin Transformer* para essa tarefa, aliada ao *fine-tuning* do modelo com o pequeno conjunto de treino, se mostrou bastante eficaz nesta tarefa de classificação de insetos adultos comparada a estratégias com extratores baseados em RNC e utilização de meta-aprendizado na etapa de treino.

Além disso, com esse mesmo conjunto de dados, foi possível chegar até um limite de 97,03% fazendo testes *three-way five-shot* com classes que já foram vistas no conjunto de treinamento, porém utilizando apenas 5 imagens de referência. Essa abordagem pode ser altamente eficaz em situações em que já existem algumas imagens dessa classe publicamente disponíveis, sendo possível fazer um *fine-tuning* do modelo com essas imagens, para depois utilizar menos imagens ainda, podendo até mesmo ser apenas uma, em um contexto específico, para classificar insetos neste contexto com classificadores *one-shot*. Utilizando um classificador *three-way one-shot*, foi possível alcançar uma acurácia de 93,70% para classificar insetos adultos, o que se mostra ótimo e prático em situações do mundo real.

Ao conduzir testes análogos com o conjunto de dados de insetos em seus estágios iniciais de vida, foi possível observar bons resultados, embora inferiores aos obtidos com as classes adultas. Grande parte dessa queda no resultado pode ser atribuída ao fato de que as espécies no estágio inicial são menos distinguíveis. Foi possível observar com a *Swin Transformer* e o *fine-tuning* do modelo, um resultado muito semelhante ao que foi visto com meta-aprendizado e RNCs. Portanto, foi possível observar que a arquitetura *Swin Transformer* não conseguiu se mostrar melhor que RNCs para a classificação dessas espécies nessas etapas.

Em última análise, é possível afirmar que a arquitetura *Swin Transformer* se destaca como uma excelente ferramenta para a classificação de insetos, mesmo diante da escassez de imagens, especialmente quando se trata de estágios adultos de vida, se mostrando uma ótima alternativa com relação às redes neurais convolucionais. Sua aplicação representa uma valiosa contribuição para a agricultura de precisão e demais aplicações no campo, promovendo avanços notáveis.

Para futuros trabalhos, seria relevante realizar estudos direcionados à otimização da acurácia nas classes de estágio inicial. Isso pode envolver investigações mais aprofundadas sobre as características específicas dos insetos nesses estágios, bem como o desenvolvimento de estratégias de treinamento mais especializadas para lidar com essas particularidades. A compreensão aprimorada dos estágios iniciais pode contribuir significativamente para a melhoria geral do desempenho do modelo.

Além disso, é recomendável conduzir estudos com um número mais expressivo de classes, incorporando uma variedade maior de insetos provenientes de diferentes culturas de plantação. Isso proporcionaria uma avaliação mais abrangente da capacidade da *Swin Transformer* em lidar com a diversidade biológica existente nos ambientes agrícolas. A

expansão do número de classes permitiria explorar a adaptabilidade do modelo a diferentes contextos e a sua capacidade de lidar com uma gama mais ampla de desafios.

Por último, dado o desempenho da *Swin Transformer* como uma eficaz extratora de características, seria pertinente conduzir experimentos voltados para a detecção de insetos nas imagens do mesmo *dataset*. Essa extensão do estudo poderia incluir a localização precisa dos insetos nas imagens, fornecendo informações valiosas para aplicações práticas, como identificação de focos de infestação e mapeamento de distribuição populacional. Esses experimentos adicionais consolidariam ainda mais a versatilidade da *Swin Transformer* como uma ferramenta abrangente para análise de imagens agrícolas.

Essas considerações indicam caminhos promissores para o avanço contínuo nesta área de pesquisa, promovendo melhorias substanciais na eficácia da *Swin Transformer* e tornando-a ainda mais robusta e aplicável em diversos cenários agrícolas.

Referências

AFAQ, S.; RAO, S. Significance of epochs on training a neural network. **International Journal of Scientific & Technology Research**, v. 9, p. 485–488, 2020. Disponível em: https://api.semanticscholar.org/CorpusID:225647672. Citado na p. 33.

ALFARISY, A.; CHEN, Q.; GUO, M. Deep learning based classification for paddy pests diseases recognition. **ICMAI '18: Proceedings of 2018 International Conference on Mathematics and Artificial Intelligence**, p. 21–25, 04 2018. Citado na p. 17.

ANGHELOF, M. M.; SUCIU, G.; CRACIUNESCU, R.; MARGHESCU, C. Intelligent system for precision agriculture. *In*: **2020 13th International Conference on Communications (COMM)**. [*S.l.: s.n.*], 2020. p. 407–410. Citado na p. 12.

BRADLEY, A. P. The use of the area under the roc curve in the evaluation of machine learning algorithms. **Pattern Recognition**, v. 30, n. 7, p. 1145–1159, 1997. ISSN 0031-3203. Disponível em: https://www.sciencedirect.com/science/article/pii/S0031320396001422. Citado na p. 36.

DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; FEI-FEI, L. Imagenet: A large-scale hierarchical image database. *In*: IEEE. **2009 IEEE conference on computer vision and pattern recognition**. [*S.l.*], 2009. p. 248–255. Citado nas pp. 16, 17, 27 e 46.

DESHMUKH, P.; SATYANARAYANA, G.; MAJHI, S.; SAHOO, U. K.; DAS, S. K. Swin transformer based vehicle detection in undisciplined traffic environment. **Expert Systems with Applications**, v. 213, p. 118992, 2023. ISSN 0957-4174. Disponível em: https://www.sciencedirect.com/science/article/pii/S0957417422020103. Citado na p. 21.

DHILLON, G. S.; CHAUDHARI, P.; RAVICHANDRAN, A.; SOATTO, S. **A Baseline for Few-Shot Image Classification**. 2020. Citado na p. 38.

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**. 2021. Citado nas pp. 20 e 25.

DUBEY, S. R.; SINGH, S. K.; CHAUDHURI, B. Activation functions in deep learning: A comprehensive survey and benchmark. **Neurocomputing**, v. 503, 07 2022. Citado na p. 29.

GOMES, J.; BORGES, D. Insect pest image recognition: A few-shot machine learning approach including maturity stages classification. **Agronomy**, v. 12, p. 1733, 07 2022. Citado nas pp. 43, 46, 47, 54 e 56.

HAMED, M.; DESOUKY, A. E. Effect of learning rate on the recognition of images. **Active and Passive Electronic Components**, v. 19, 01 1996. Citado na p. 32.

HU, H.; ZHANG, Z.; XIE, Z.; LIN, S. Local relation networks for image recognition. *In*: **Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)**. [*S.l.: s.n.*], 2019. p. 3464–3473. Citado na p. 20.

HøYE, T. T.; ÄRJE, J.; BJERGE, K.; HANSEN, O. L. P.; IOSIFIDIS, A.; LEESE, F.; MANN, H. M. R.; MEISSNER, K.; MELVAD, C.; RAITOHARJU, J. Deep learning and computer vision will transform entomology. **Proceedings of the National Academy of Sciences**, v. 118, n. 2, p. e2002545117, 2021. Citado na p. 12.

JAMES LOY. **A Comprehensive Guide to Microsoft's Swin Transformer**. 2022. Disponível em: https://towardsdatascience.com/a-comprehensive-guide-to-swin-transformer-64965f89d14c – acesso em 09 jul. 2023. Citado na p. 23.

JAY ALAMMAR. **The Illustrated Transformer**. 2018. Disponível em: http://jalammar.github.io/illustrated-transformer/ – acesso em 07 jul. 2023. Citado nas pp. 18 e 19.

KASINATHAN, T.; SINGARAJU, D.; UYYALA, S. R. Insect classification and detection in field crops using modern machine learning techniques. **Information Processing in Agriculture**, v. 8, n. 3, p. 446–457, 2021. ISSN 2214-3173. Disponível em: https://www.sciencedirect.com/science/article/pii/S2214317320302067. Citado na p. 13.

KHANRAMAKI, M.; ASLI-ARDEH, E.; KOZEGAR, E. Citrus pests classification using an ensemble of deep learning models. **Computers and Electronics in Agriculture**, v. 186, p. 106192, 07 2021. Citado na p. 15.

KIM, J.-H.; KIM, N.; WON, C. S. Facial expression recognition with swin transformer. **ArXiv**, abs/2203.13472, 2022. Citado na p. 21.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **CoRR**, abs/1412.6980, 2014. Citado na p. 31.

LI, W.; ZHENG, T.; YANG, Z.; LI, M.; SUN, C.; YANG, X. Classification and detection of insects from field images using deep learning for smart pest management: A systematic review. **Ecological Informatics**, v. 66, p. 101460, 2021. ISSN 1574-9541. Disponível em: https://www.sciencedirect.com/science/article/pii/S157495412100251X. Citado nas pp. 12 e 13.

LI, W.; ZHENG, T.; YANG, Z.; LI, M.; SUN, C.; YANG, X. Classification and detection of insects from field images using deep learning for smart pest management: A systematic review. **Ecological Informatics**, v. 66, p. 101460, 10 2021. Citado na p. 16.

LIANG, D.; CHEN, X.; XU, W.; ZHOU, Y.; BAI, X. Transcrowd: weakly-supervised crowd counting with transformers. **Science China Information Sciences**, Springer, v. 65, n. 6, p. 1–14, 2022. Citado na p. 21.

LIANG, J.; CAO, J.; SUN, G.; ZHANG, K.; GOOL, L. V.; TIMOFTE, R. Swinir: Image restoration using swin transformer. **arXiv preprint arXiv:2108.10257**, 2021. Citado na p. 21.

LIN, A.; CHEN, B.; XU, J.; ZHANG, Z.; LU, G.; ZHANG, D. Ds-transunet: Dual swin transformer u-net for medical image segmentation. **IEEE Transactions on Instrumentation and Measurement**, v. 71, p. 1–15, 2022. Citado na p. 21.

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. Citado nas pp. 16 e 17.

LIU, J.; WANG, X. Tomato diseases and pests detection based on improved yolo v3 convolutional neural network. **Frontiers in Plant Science**, v. 11, p. 898, 06 2020. Citado na p. 13.

LIU, Z.; GAO, J.; YANG, G.; ZHANG, H.; HE, Y. Localization and classification of paddy field pests using a saliency map and deep convolutional neural network. **Scientific Reports**, v. 6, p. 20410, 02 2016. Citado na p. 17.

LIU, Z.; HU, H.; LIN, Y.; YAO, Z.; XIE, Z.; WEI, Y.; NING, J.; CAO, Y.; ZHANG, Z.; DONG, L.; WEI, F.; GUO, B. Swin transformer v2: Scaling up capacity and resolution. *In*: **International Conference on Computer Vision and Pattern Recognition (CVPR)**. [*S.l.: s.n.*], 2022. Citado na p. 25.

LIU, Z.; LIN, Y.; CAO, Y.; HU, H.; WEI, Y.; ZHANG, Z.; LIN, S.; GUO, B. Swin transformer: Hierarchical vision transformer using shifted windows. *In*: **Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)**. [*S.l.: s.n.*], 2021. Citado nas pp. 13, 16, 21 e 25.

MAHADEVKAR, S.; KHEMANI, B.; PATIL, S.; KOTECHA, K.; VORA, D.; ABRAHAM, A.; GABRALLA, L. A review on machine learning styles in computer vision - techniques and future directions. **IEEE Access**, PP, p. 1–1, 01 2022. Citado na p. 26.

MICROSOFT. **Swin Transformer**. [*S.l.*]: GitHub, 2021. https://github.com/SwinTransformer. Citado na p. 45.

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. **Foundations of Machine Learning**. Cambridge, MA, USA: MIT Press, 2018. Citado na p. 27.

OSMAN, H.; BLOSTEIN, S. Backpropagation algorithm for multiresolution image classification. *In*: **Proceedings 1999 International Conference on Image Processing (Cat. 99CH36348)**. [*S.l.: s.n.*], 1999. v. 1, p. 519–523 vol.1. Citado na p. 31.

RADIUK, P. Impact of training set batch size on the performance of convolutional neural networks for diverse datasets. **Information Technology and Management Science**, v. 20, p. 20–24, 12 2017. Citado na p. 32.

RADOSAVOVIC, I.; KOSARAJU, R. P.; GIRSHICK, R.; HE, K.; DOLLáR, P. **Designing Network Design Spaces**. 2020. Citado na p. 25.

REDMON, J.; FARHADI, A. **YOLOv3: An Incremental Improvement**. 2018. Citado na p. 17.

RUBY, U.; YENDAPALLI, V. Binary cross entropy with deep learning technique for image classification. **International Journal of Advanced Trends in Computer Science and Engineering**, v. 9, 10 2020. Citado na p. 30.

SNELL, J.; SWERSKY, K.; ZEMEL, R. S. **Prototypical Networks for Few-shot Learning**. 2017. Citado nas pp. 38 e 39.

THENMOZHI, K.; Srinivasulu Reddy, U. Crop pest classification based on deep convolutional neural network and transfer learning. **Computers and Electronics in Agriculture**, v. 164, p. 104906, 2019. ISSN 0168-1699. Disponível em: https://www.sciencedirect.com/science/article/pii/S0168169919310695. Citado na p. 13.

TIAN, Y.; ZHANG, Y.; ZHANG, H. Recent advances in stochastic gradient descent in deep learning. **Mathematics**, v. 11, n. 3, 2023. ISSN 2227-7390. Disponível em: https://www.mdpi.com/2227-7390/11/3/682. Citado na p. 31.

TOUVRON, H.; CORD, M.; DOUZE, M.; MASSA, F.; SABLAYROLLES, A.; JEGOU, H. Training data-efficient image transformers amp; distillation through attention. *In*: MEILA, M.; ZHANG, T. (Ed.). **Proceedings of the 38th International Conference on Machine Learning**. [*S.l.*]: PMLR, 2021. (Proceedings of Machine Learning Research, v. 139), p. 10347–10357. Citado na p. 25.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. **Attention Is All You Need**. 2017. Citado na p. 17.

VINYALS, O.; BLUNDELL, C.; LILLICRAP, T.; KAVUKCUOGLU, k.; WIERSTRA, D. Matching networks for one shot learning. *In*: LEE, D.; SUGIYAMA, M.; LUXBURG, U.; GUYON, I.; GARNETT, R. (Ed.). **Advances in Neural Information Processing Systems**. [*S.l.*]: Curran Associates, Inc., 2016. v. 29. Citado na p. 38.

WANG, J.; LI, Y.; FENG, H.; REN, L.; DU, X.; WU, J. Common pests image recognition based on deep convolutional neural network. **Computers and Electronics in Agriculture**, v. 179, p. 105834, 12 2020. Citado na p. 16.

WANG, Y.; YAO, Q.; KWOK, J. T.; NI, L. M. Generalizing from a few examples: A survey on few-shot learning. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 53, n. 3, jun 2020. ISSN 0360-0300. Disponível em: https://doi.org/10.1145/3386252. Citado na p. 37.

WU, X.; ZHAN, C.; LAI, Y.; CHENG, M.-M.; YANG, J. Ip102: A large-scale benchmark dataset for insect pest recognition. *In*: **IEEE CVPR**. [*S.l.: s.n.*], 2019. p. 8787–8796. Citado nas pp. 17, 42, 44, 53 e 59.

XIE, C.; WANG, R.; ZHANG, J.; CHEN, P.; DONG, W.; LI, R.; CHEN, T.; CHEN, H. Multi-level learning features for automatic classification of field crop pests. **Computers and Electronics in Agriculture**, Elsevier, v. 152, p. 233–241, 2018. Citado na p. 17.

YANG, J.; GUO, X.; LI, Y.; MARINELLO, F.; ERCISLI, S.; ZHANG, Z. A survey of few-shot learning in smart agriculture: developments, applications, and challenges. **Plant Methods**, v. 18, 03 2022. Citado na p. 38.

Anexos

Anexo A – Tabela de distribuição de classes do IP102

Tabela A.10 – Quantidade de imagens por classe no *dataset* IP102. O nome das classes foi mantido da forma que foi divulgada pelos autores do conjunto de dados.

Classe	**Total de Imagens**	**Classe**	**Total de Imagens**
rice leaf roller	1115	*rice leaf caterpillar*	487
paddy stem maggot	261	*asiatic rice borer*	1053
yellow rice borer	504	*rice gall midge*	506
Rice Stemfly	369	*brown plant hopper*	834
white backed plant hopper	893	*small brown plant hopper*	553
rice water weevil	856	*rice leafhopper*	404
grain spreader thrips	173	*rice shell pest*	409
grub	860	*mole cricket*	1649
wireworm	887	*white margined moth*	147
black cutworm	854	*large cutworm*	491
yellow cutworm	479	*red spider*	529
corn borer	1698	*army worm*	1071
aphids	4094	*Potosiabre vitarsis*	565
peach borer	691	*english grain aphid*	654
green bug	325	*bird cherry-oataphid*	708
wheat blossom midge	485	*penthaleus major*	233
longlegged spider mite	245	*wheat phloeothrips*	292
wheat sawfly	339	*cerodonta denticornis*	137
beet fly	104	*flea beetle*	789
cabbage army worm	855	*beet army worm*	1604
Beet spot flies	299	*meadow moth*	270
beet weevil	308	*sericaorient alismots chulsky*	191
alfalfa weevil	524	*flax budworm*	1066
alfalfa plant bug	656	*tarnished plant bug*	820
Locustoidea	1394	*lytta polita*	654
legume blister beetle	1409	*blister beetle*	1897
therioaphis maculata Buckton	260	*odontothrips loti*	177
Thrips	879	*alfalfa seed chalcid*	185

Pieris canidia	469	*Apolygus lucorum*	381
Limacodidae	1400	*Viteus vitifoliae*	354
Colomerus vitis	176	*Brevipoalpus lewisi McGregor*	79
oides decempunctata	283	*Polyphagotars onemus latus*	84
Pseudococcus comstocki Kuwana	307	*parathrene regalis*	140
Ampelophaga	764	*Lycorma delicatula*	5310
Xylotrechus	1150	*Cicadella viridis*	1279
Miridae	5081	*Trialeurodes vaporariorum*	692
Erythroneura apicalis	71	*Papilio xuthus*	449
Panonchus citri McGregor	385	*Phyllocoptes oleiverus ashmead*	172
Icerya purchasi Maskell	722	*Unaspis yanonensis*	419
Ceroplastes rubens	258	*Chrysomphalus aonidum*	225
Parlatoria zizyphus Lucus	74	*Nipaecoccus vastalor*	99
Aleurocanthus spiniferus	690	*Tetradacus c Bactrocera minax*	388
Dacus dorsalis(Hendel)	439	*Bactrocera tsuneonis*	168
Prodenia litura	1304	*Adristyrannus*	311
Phyllocnistis citrella Stainton	404	*Toxoptera citricidus*	189
Toxoptera aurantii	226	*Aphis citricola Vander Goot*	351
Scirtothrips dorsalis Hood	807	*Dasineura sp*	505
Lawana imitata Melichar	578	*Salurnis marginella Guerr*	487
Deporaus marginatus Pascoe	210	*Chlumetia transversa*	305
Mango flat beak leafhopper	93	*Rhytidodera bowrinii white*	555
Sternochetus frigidus	458	*Cicadellidae*	5740

Anexo B – Tabela de distribuição de classes do IP-FSL

Tabela B.11 – Quantidade de imagens por classe adulta e de estágio inicial no *dataset* IP-FSL. O nome das classes foi mantido da forma que foi divulgada pelos autores do IP102.

Classe	**Estágio Adulto / Inicial**	**Classe**	**Estágio Adulto / Inicial**
rice leaf roller	50/50	*rice leaf caterpillar*	50/50
paddy stem maggot	50/50	*asiatic rice borer*	50/50
yellow rice borer	50/50	*rice gall midge*	50/31
Rice Stemfly	50/47	*brown plant hopper*	50/17
white backed plant hopper	50/18	*small brown plant hopper*	50/-
rice water weevil	50/50	*rice leafhopper*	50/-
grain spreader thrips	50/-	*rice shell pest*	50/50
grub	-/50	*mole cricket*	50/-
wireworm	50/50	*white margined moth*	26/50
black cutworm	50/50	*large cutworm*	50/50
yellow cutworm	50/50	*red spider*	50/-
corn borer	50/50	*army worm*	35/50
aphids	50/-	*Potosiabre vitarsis*	50/-
peach borer	50/50	*english grain aphid*	50/-
green bug	50/-	*bird cherry-oataphid*	50/-
wheat blossom midge	50/50	*penthaleus major*	50/-
longlegged spider mite	50/-	*wheat phloeothrips*	50/-
wheat sawfly	50/50	*cerodonta denticornis*	50/32
beet fly	50/-	*flea beetle*	50/-
cabbage army worm	50/50	*beet army worm*	50/50
Beet spot flies	50/50	*meadow moth*	50/25
beet weevil	50/-	*sericaorient alismots chulsky*	50/-
alfalfa weevil	50/50	*flax budworm*	50/50
alfalfa plant bug	50/-	*tarnished plant bug*	50/-
Locustoidea	50/-	*lytta polita*	50/-
legume blister beetle	50/-	*blister beetle*	50/-
therioaphis maculata Buckton	50/-	*odontothrips loti*	50/-

Thrips	50/-	*alfalfa seed chalcid*	50/-
Pieris canidia	50/-	*Apolygus lucorum*	50/-
Limacodidae	50/50	*Viteus vitifoliae*	-/50
Colomerus vitis	-/50	*Brevipoalpus lewisi McGregor*	47/-
oides decempunctata	50/-	*Polyphagotars onemus latus*	50/-
Pseudococcus comstocki Kuwana	50/-	*parathrene regalis*	40/30
Ampelophaga	50/50	*Lycorma delicatula*	50/-
Xylotrechus	50/-	*Cicadella viridis*	50/-
Miridae	50/-	*Trialeurodes vaporariorum*	50/-
Erythroneura apicalis	42/-	*Papilio xuthus*	50/50
Panonchus citri McGregor	50/-	*Phyllocoptes oleiverus ashmead*	-/50
Icerya purchasi Maskell	50/-	*Unaspis yanonensis*	50/-
Ceroplastes rubens	50/-	*Chrysomphalus aonidum*	50/-
Parlatoria zizyphus Lucus	44/-	*Nipaecoccus vastalor*	50/-
Aleurocanthus spiniferus	-/50	*Tetradacus c Bactrocera minax*	50/50
Dacus dorsalis(Hendel)	50/40	*Bactrocera tsuneonis*	50/20
Prodenia litura	50/50	*Adristyrannus*	50/40
Phyllocnistis citrella Stainton	50/50	*Toxoptera citricidus*	50/-
Toxoptera aurantii	50/-	*Aphis citricola Vander Goot*	50/-
Scirtothrips dorsalis Hood	50/-	*Dasineura sp*	33/50
Lawana imitata Melichar	50/-	*Salurnis marginella Guerr*	50/-
Deporaus marginatus Pascoe	50/-	*Chlumetia transversa*	50/50
Mango flat beak leafhopper	50/-	*Rhytidodera bowrinii white*	50/-
Sternochetus frigidus	50/-	*Cicadellidae*	50/-

Anexo C – Tabela de Resultados por classe

Tabela C.12 – Resultados e métricas obtidas por cada uma das 102 classes do IP102 pelo modelo SwinL com *batch* de 64 imagens. O nome das classes foi mantido da forma que foi divulgada pelos autores do IP102. O total de imagens é referente a soma das imagens de teste de cada um dos 10 folds.

Classe	**Total de Imagens**	**VP**	**VN**	**FP**	**FN**	**Precisão**	**Recall**	**F1-Score**
rice leaf roller	110	70	7216	57	40	0,55	0,64	0,59
rice leaf caterpillar	47	9	7314	22	38	0,29	0,19	0,23
paddy stem maggot	24	4	7350	9	20	0,31	0,17	0,22
asiatic rice borer	104	57	7221	58	47	0,50	0,55	0,52
yellow rice borer	50	24	7307	26	26	0,48	0,48	0,48
rice gall midge	50	33	7313	20	17	0,62	0,66	0,64
Rice Stemfly	36	9	7330	17	27	0,35	0,25	0,29
brown plant hopper	83	29	7241	59	54	0,33	0,35	0,34
white backed plant hopper	88	28	7225	70	60	0,29	0,32	0,30
small brown plant hopper	54	19	7303	26	35	0,42	0,35	0,38
rice water weevil	84	58	7267	32	26	0,64	0,69	0,67
rice leafhopper	40	10	7334	9	30	0,53	0,25	0,34
grain spreader thrips	16	10	7360	7	6	0,59	0,62	0,61
rice shell pest	40	15	7316	27	25	0,36	0,38	0,37
grub	84	75	7289	10	9	0,88	0,89	0,89
mole cricket	163	162	7217	3	1	0,98	0,99	0,99
wireworm	87	72	7276	20	15	0,78	0,83	0,80
white margined moth	13	3	7365	5	10	0,38	0,23	0,29
black cutworm	84	48	7251	48	36	0,50	0,57	0,53
large cutworm	47	10	7311	25	37	0,29	0,21	0,24
yellow cutworm	46	10	7312	25	36	0,29	0,22	0,25
red spider	51	32	7310	22	19	0,59	0,63	0,61
corn borer	169	120	7138	76	49	0,61	0,71	0,66
army worm	106	51	7221	56	55	0,48	0,48	0,48

aphids	407	322	6826	150	85	0,68	0,79	0,73
Potosiabre vitarsis	55	48	7323	5	7	0,91	0,87	0,89
peach borer	67	43	7294	22	24	0,66	0,64	0,65
english grain aphid	64	19	7280	39	45	0,33	0,30	0,31
green bug	31	5	7334	18	26	0,22	0,16	0,19
bird cherry-oataphid	70	26	7272	41	44	0,39	0,37	0,38
wheat blossom midge	47	42	7329	7	5	0,86	0,89	0,88
penthaleus major	22	10	7356	5	12	0,67	0,45	0,54
longlegged spider mite	23	6	7346	14	17	0,30	0,26	0,28
wheat phloeothrips	27	13	7343	13	14	0,50	0,48	0,49
wheat sawfly	33	17	7341	9	16	0,65	0,52	0,58
cerodonta denticornis	13	2	7361	9	11	0,18	0,15	0,17
beet fly	10	3	7369	4	7	0,43	0,30	0,35
flea beetle	77	66	7299	7	11	0,90	0,86	0,88
cabbage army worm	84	35	7244	55	49	0,39	0,42	0,40
beet army worm	160	74	7115	108	86	0,41	0,46	0,43
Beet spot flies	29	14	7339	15	15	0,48	0,48	0,48
meadow moth	26	11	7350	7	15	0,61	0,42	0,50
beet weevil	30	16	7342	11	14	0,59	0,53	0,56
sericaorient alismots chulsky	17	9	7364	2	8	0,82	0,53	0,64
alfalfa weevil	51	31	7317	15	20	0,67	0,61	0,64
flax budworm	105	38	7198	80	67	0,32	0,36	0,34
alfalfa plant bug	64	27	7301	18	37	0,60	0,42	0,50
tarnished plant bug	81	32	7280	22	49	0,59	0,40	0,47
Locustoidea	137	134	7241	5	3	0,96	0,98	0,97
lytta polita	64	27	7295	24	37	0,53	0,42	0,47
legume blister beetle	140	87	7197	46	53	0,65	0,62	0,64
blister beetle	188	128	7118	77	60	0,62	0,68	0,65
therioaphis maculata Buckton	24	0	7342	17	24	0,00	0,00	0,00
odontothrips loti	16	4	7356	11	12	0,27	0,25	0,26
Thrips	86	32	7245	52	54	0,38	0,37	0,38
alfalfa seed chalcid	17	5	7358	8	12	0,38	0,29	0,33
Pieris canidia	46	44	7335	2	2	0,96	0,96	0,96
Apolygus lucorum	36	9	7335	12	27	0,43	0,25	0,32
Limacodidae	139	129	7233	11	10	0,92	0,93	0,92

Viteus vitifoliae	34	23	7337	12	11	0,66	0,68	0,67
Colomerus vitis	16	7	7361	6	9	0,54	0,44	0,48
Brevipoalpus lewisi McGregor	6	0	7377	0	6	0,00	0,00	0,00
oides decempunctata	26	25	7355	2	1	0,93	0,96	0,94
Polyphagotars onemus latus	7	2	7375	1	5	0,67	0,29	0,40
Pseudococcus comstocki Kuwana	30	13	7342	11	17	0,54	0,43	0,48
parathrene regalis	13	3	7369	1	10	0,75	0,23	0,35
Ampelophaga	75	58	7295	13	17	0,82	0,77	0,79
Lycorma delicatula	530	471	6735	118	59	0,80	0,89	0,84
Xylotrechus	114	96	7256	13	18	0,88	0,84	0,86
Cicadella viridis	126	87	7223	34	39	0,72	0,69	0,70
Miridae	506	448	6774	103	58	0,81	0,89	0,85
Trialeurodes vaporariorum	67	52	7302	14	15	0,79	0,78	0,78
Erythroneura apicalis	6	2	7374	3	4	0,40	0,33	0,36
Papilio xuthus	43	39	7340	0	4	1,00	0,91	0,95
Panonchus citri McGregor	37	27	7333	13	10	0,68	0,73	0,70
Phyllocoptes oleiverus ashmead	16	11	7359	8	5	0,58	0,69	0,63
Icerya purchasi Maskell	71	57	7295	17	14	0,77	0,80	0,79
Unaspis yanonensis	41	25	7329	13	16	0,66	0,61	0,63
Ceroplastes rubens	24	16	7357	2	8	0,89	0,67	0,76
Chrysomphalus aonidum	21	12	7358	4	9	0,75	0,57	0,65
Parlatoria zizyphus Lucus	6	2	7377	0	4	1,00	0,33	0,50
Nipaecoccus vastalor	9	1	7370	4	8	0,20	0,11	0,14
Aleurocanthus spiniferus	67	58	7304	12	9	0,83	0,87	0,85
Tetradacus c Bactrocera minax	37	22	7331	15	15	0,59	0,59	0,59
Dacus dorsalis(Hendel)	43	34	7329	11	9	0,76	0,79	0,77

Bactrocera tsuneonis	16	4	7359	8	12	0,33	0,25	0,29
Prodenia litura	130	69	7183	70	61	0,50	0,53	0,51
Adristyrannus	30	16	7348	5	14	0,76	0,53	0,63
Phyllocnistis citrella Stainton	40	30	7337	6	10	0,83	0,75	0,79
Toxoptera citricidus	17	5	7361	5	12	0,50	0,29	0,37
Toxoptera aurantii	21	7	7352	10	14	0,41	0,33	0,37
Aphis citricola Vander Goot	34	10	7336	13	24	0,43	0,29	0,35
Scirtothrips dorsalis Hood	80	48	7265	38	32	0,56	0,60	0,58
Dasineura sp	50	22	7312	21	28	0,51	0,44	0,47
Lawana imitata Melichar	56	44	7313	14	12	0,76	0,79	0,77
Salurnis marginella Guerr	47	28	7323	13	19	0,68	0,60	0,64
Deporaus marginatus Pascoe	20	10	7360	3	10	0,77	0,50	0,61
Chlumetia transversa	30	11	7338	15	19	0,42	0,37	0,39
Mango flat beak leafhopper	8	0	7375	0	8	0,00	0,00	0,00
Rhytidodera bowrinii white	54	35	7313	16	19	0,69	0,65	0,67
Sternochetus frigidus	44	23	7326	13	21	0,64	0,52	0,57
Cicadellidae	573	508	6705	105	65	0,83	0,89	0,86