Aplicação de Data Augmentation em Redes Convolucionais para detecção do Cancro Europeu das Pomáceas

Ântony de Aguiar Salvi[1], **Paulo C Camargo Jr**[1],
Wilson Castello Branco Neto[1], **Andrey de Aguiar Salvi**[2]

[1]Instituto Federal de Santa Catarina (IFSC)
Rua Heitor Vila Lobos, 225 - São Francisco - 88506-400 - Lages - SC - Brasil

[2]Lojas Renner S.A
Rua Joaquim Pôrto Vilanova, 401 Jardim do Salso - 91410-400 - Porto Alegre - RS - Brasil

{antony.s, paulo.ccj}@aluno.ifsc.edu.br

wilson.castello@ifsc.edu.br

andrey.salvi@lojasrenner.com.br

Abstract. *European canker affects orchards from several regions and emerged in Brazil in Rio Grande do Sul spreading to other states that represent a large portion of the production of apple. The disease is a threat due to its rapid spread and for the difficulty to detect it. In order to solve this problem, a CNN was proposed for disease classification. However, the lack of images makes training very difficult and therefore this work attempts to overcome this problem by applying Data Augmentation techniques. However, due to the small amount of images and the nature of the problem, the desired results were not obtained. The result in validation set for F-Beta was 0.842, obtained by VGG19 with combined DA techniques.*

Resumo. *O cancro europeu afeta os pomares de diversas regiões e surgiu no Brasil no Rio Grande do Sul espalhando-se em outros estados que representam grande parcela da produção de maçã. A doença é uma ameaça por sua rápida propagação e pela dificuldade na sua detecção. Para solucionar este problema foi proposta uma CNN para classificação da doença. Entretanto, a falta de imagens torna o treinamento muito complicado e por isso este trabalho tenta contornar este problema aplicando técnicas de Data Augmentation. No entanto, devido a baixa quantidade de imagens e a natureza do problema, os resultados desejados não puderam ser obtidos. O resultado no conjunto de validação para o F-Beta foi de 0.842, obtido pelo VGG19 com técnicas de DA combinadas.*

1. Introdução

O cancro europeu das pomáceas é uma doença que afeta os pomares de diversas regiões do mundo (Weber, 2014). O primeiro relato que se tem da doença no Brasil foi em um viveiro no estado do Rio Grande do Sul em 2002, espalhando-se não só em grande parte do estado, como também nos estados de Santa Catarina e Paraná. Somados, esses estados representam toda a produção nacional da fruta (Kist, 2019). Causada pelo fungo *Neonectria ditissima*, a doença representa uma grave ameaça se não combatida devido a sua alta

capacidade de propagação e ao fato dos cultivos de macieiras estarem em locais muito próximos uns dos outros, o que facilita a sua disseminação, causando grandes prejuízos aos proprietários (Alves et al., 2019).

Para identificar a possível presença do fungo é necessária uma inspeção por fitopatologistas experientes através de uma amostra da planta. Esta tarefa, normalmente, é feita pelos fitopatologistas da Epagri ou da Embrapa, o que acaba dificultando o diagnóstico, pois nem todos os agricultores se encontram em regiões próximas a estas empresas e a etapa de deslocamento torna-se um empecilho. A Figura 1 apresenta duas plantas com lesões geradas pelo cancro europeu, uma delas no tronco (a) e a outra em um ramo (b).

Figura 1. Doença do cancro europeu em diferentes partes da árvore.
Fonte: Imagem obtida e cedida pelos Fitopatologistas da Epagri.

Para conscientizar os produtores, auxiliar na detecção da doença e acelerar o diagnóstico, foi criada a plataforma Cancontrol que é composta por um aplicativo móvel e um sistema móvel. Uma de suas funcionalidades presente nos dois sistemas, é uma seção composta por textos, imagens e vídeos sobre o cancro europeu. Há também um módulo de monitoramento no aplicativo móvel, por meio do qual o produtor pode cadastrar um monitoramento com dados sobre uma planta específica, além de realizar o *upload* de imagens para posterior análise de um especialista da Epagri. A disponibilização da plataforma em 2021 facilitou o diagnóstico da doença porque os produtores não precisam mais levar as amostras da planta presencialmente na Epagri, mas apesar de as imagens serem enviadas por meio do aplicativo do Cancontrol, a análise para determinar se há indícios da doença pode levar algumas semanas, pois ainda depende de análise humana (Branco Neto et al., 2021).

Para reduzir o número de análises manuais e diminuir a sobrecarga dos fitopatologistas da Epagri, foi desenvolvida uma Rede Neural Convolucional (CNN - *Convolutional Neural Network*) para processar as imagens e obter resultados automatizados. No trabalho realizado por (Mattos e Ribeiro, 2022), foram utilizadas arquiteturas tradicionais, visto que elas trazem uma maior agilidade no desenvolvimento do modelo, possibilitando o uso de redes pré-treinadas. As métricas de acurácia, Precisão, e F-Score foram utilizadas para avaliar a eficiência dos diversos modelos. Dos nove (9) modelos treinados, o que

obteve o melhor resultado foi o VGG, com F-Beta de 0.877, Precisão de 0.909 e Recall igual a 0.769.

Apesar de promissores, esses resultados possuem limitações devido à baixa quantidade de imagens disponíveis para treinamento, teste e validação do modelo, o que pode resultar em uma rede com alto *overfitting*[1]. Além disso, alguns modelos obtiveram pontuações inesperadas, como o Mobilenet V2 e Alexnet, que obtiveram melhores resultados que o Mobilenet V3. Normalmente isso não ocorre em experimentos relatados na literatura da área, o que reforça a hipótese de *overfitting* (Howard et al., 2017) (Howard et al., 2019).

Em função do exposto, pode-se estabelecer a seguinte questão de pesquisa: A aplicação de técnicas de Data Augmention[2] (DA) é capaz de melhorar os resultados gerados por estes ou por outros modelos treinados para o diagnóstico do cancro europeu das pomáceas, reduzindo a possibilidade de *overfitting*?

Com o intuito de obter resultados mais precisos, para posteriormente ativar a Rede Neural Convolucional na plataforma Cancontrol, este trabalho tem como objetivo melhorar a assertividade do modelo para permitir que os resultados sejam instantâneos. Para atingir este objetivo, foram definidos os seguintes objetivos específicos:

- Implementar novos modelos não testados no trabalho de Mattos e Ribeiro (2022);
- Aplicar estratégias de *data augmentation* para aumentar artificialmente a diversidade dos dados dentro do *dataset*;
- Avaliar os novos modelos e comparar os resultados com os anteriores.

Este trabalho está dividido em quatro etapas, a primeira etapa corresponde ao estudo do atual modelo e de outros modelos alternativos não testados no trabalho anterior. Na segunda etapa foram implementados os novos modelos. A terceira etapa consiste em utilizar *data augmentation* para gerar novas imagens. Na quarta etapa foram testados os novos modelos e comparados os resultados obtidos.

Os modelos utilizados por Mattos e Ribeiro (2022) foram avaliados, assim como os resultados obtidos. Após, foi feito um levantamento bibliográfico para conhecer outros modelos, que também foram implementados.

Em seguida foi realizado o *Data Augmentation* que consiste em criar imagens artificiais. As imagens são geradas a partir de pequenas alterações no conjunto original. Uma das formas possíveis, por exemplo, é a utilização de transformações geométricas, como aplicar uma translação, rotacionar etc. Essas alterações criam versões ligeiramente modificadas das imagens originais o que diversifica o treinamento do modelo. Por fim, foram avaliados os resultados obtidos e comparados com os anteriores para gerar as conclusões.

Quanto à natureza deste trabalho, ele se classifica como uma pesquisa aplicada quantitativa quanto a sua abordagem, explicativa quanto aos seus objetivos e bibliográfica e experimental quanto aos seus procedimentos técnicos.

[1] Um modelo com *overfitting* não consegue generalizar o problema de forma adequada. Ele aprende a classificar apenas os dados de treino. A generalização consiste na capacidade do modelo fornecer as respostas corretas, em dados não vistos antes (não utilizados no treino ou na validação).

[2] *Data Augmentation* é a técnica de criar imagens artificiais a partir de imagens reais.

O artigo é constituído de cinco seções, na segunda seção é apresentado o referencial teórico. Na terceira seção é apresentada a implementação dos modelos escolhidos. Na quarta seção são avaliados os resultados obtidos de cada modelo e a efetividade das técnicas utilizadas. Na quinta seção são apresentadas as conclusões.

2. Referencial Teórico

Nas subseções 2.1 e 2.2 são apresentados os conceitos essenciais para o entendimento das Redes Neurais Artificiais e Redes Neurais Convolucionais, respectivamente, e na seção 2.3 são apresentadas as principais arquiteturas de Redes Neurais Convolucionais. Na subseção 2.4 são apresentadas algumas técnicas de *Data Augmentation* e, por fim, em 2.5 são apresentados alguns trabalhos similares.

2.1. Redes Neurais Artificiais

As Redes Neurais Artificiais (ANN - *Artificial Neural Network*) são um subconjunto de técnicas da área de *Machine Learning* que caracterizam-se por imitar, de forma superficial, o funcionamento de redes de neurônios humanos. Uma ANN é composta de diversas camadas. A camada inicial, que possui os dados de entrada, é chamada de *Input Layer*, a camada final, que fornece os resultados, é chamada de *Output Layer* e as camadas intermediárias são chamadas de *Hidden Layers*.

Cada camada, por sua vez, é composta de diversos nodos ou neurônios (unidades). Um nodo é composto por ligações de entrada que servem para alimentar a função de entrada do neurônio. Para cada valor de entrada existe um peso associado, que determina a força e o sinal de conexão. Os valores, com seus respectivos pesos, passam por uma função de entrada, denotada pelo símbolo de somatório na Figura 2. Ela tem como objetivo resumir os valores para uma representação simplificada. O resultado dessa função é então utilizado na função de ativação que faz o processamento através de uma função, geralmente, não linear. Após a função de ativação, os valores resultantes são distribuídos para as ligações de saída que alimentam os nodos das camadas futuras.

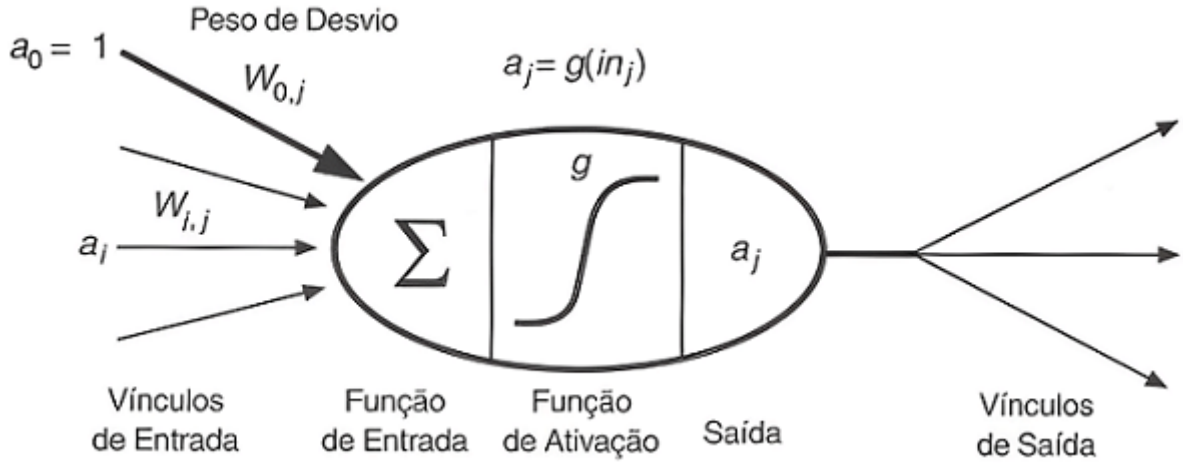


Figura 2. Exemplo de neurônio
Fonte: Russell e Norvig (2022)

Na Figura 2 é possível perceber as ligações de entrada a_i que partem da unidade i para a unidade j, com pesos w_{ij}, e são enviados para função de entrada $\sum_{i=0}^{n} a_i w_{ij}$. O

resultado da função de entrada é então processado com a função de ativação. Exemplos de funções de ativação são a sigmoide, softmax, ReLU e tahn.

A função sigmoide possui a fórmula apresentada na Equação (1) e ela fornece valores dentro do intervalo $[0, 1]$, sendo utilizada para classificação binária:

$$\alpha(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

A função softmax, apresentada na Equação (2), é uma generalização da sigmoide utilizada para problemas que envolvam mais do que duas variáveis para classificação.

$$\sigma(x) = \frac{e^x}{\sum_{j=1}^{K} e_j^x} \tag{2}$$

A função ReLU, apresentada na Equação (3), mantém o valor original da função de entrada para os valores positivos e para os valores negativos atribui o valor 0.

$$f(x) = \begin{cases} x & \text{se } x > 0 \\ 0 & \text{se } x < 0 \end{cases} \tag{3}$$

A função tahn, apresentada na Equação (4), retorna um valor no intervalo [-1, 1]. Ela, na verdade, é uma versão aumentada e transladada da função sigmoide.

$$Tahn(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{4}$$

As ANN podem ser do tipo recorrentes ou do tipo propagação direta. Redes recorrentes utilizam os resultados de seus neurônios para retroalimentar a rede, formando um ciclo. Redes de propagação direta possuem várias camadas, são lineares e nenhum neurônio é reutilizado.

A Figura 3 apresenta uma rede do tipo propagação direta. A camada de entrada representa as imagens de entrada, seguida por duas (2) camadas ocultas, uma camada de saída e a camada de perda ou camada de custo, que é utilizada para calcular a diferença entre os resultados da rede e os valores corretos dos dados utilizados no treinamento. A camada de saída é responsável pela classificação dos dados, que representam os resultados gerados pela rede.

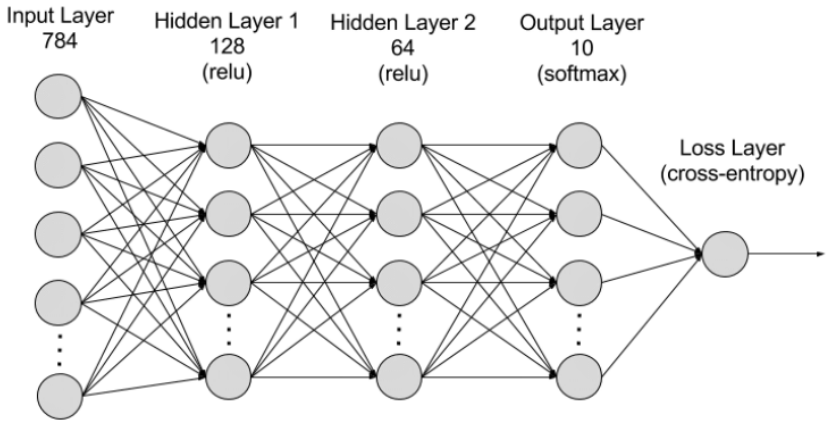


Figura 3. Uma rede do tipo propagação direta
Fonte: Amazon (2023)

Os algoritmos de propagação direta presentes neste trabalho realizam sua aprendizagem por retropropagação de erro (*backpropagation*) (Haykin, 2009). Durante o treinamento, após o algoritmo gerar uma saída é iniciado o processo de aprendizagem do modelo *backpropagation*. Nesta etapa a rede é percorrida no sentido inverso e seus pesos são atualizados de acordo com a regra de correção de erro.

Utilizando uma função de erro/custo (*loss function*), é possível saber o quão distante a resposta gerada pelo modelo está em comparação com a saída real. Ao aplicar a derivada da *loss function*, utilizando a regra da cadeia, é possível avaliar o erro de cada peso individualmente e se esse erro é decorrente de um valor menor ou maior que o valor correto. Com essa informação é possível saber se o valor do peso deve ser aumentado ou diminuído.

2.2. Redes Neurais Convolucionais

As CNN são um tipo de Rede Neural Artificial (ANN) com foco em processamento de dados, como imagens, vídeos e faixas de áudio, que ganharam notoriedade após o trabalho de Lecun et al. (1998). CNNs aplicam uma operação matemática chamada convolução, que é um tipo de operação linear (Goodfellow et al., 2016). Segundo Goodfellow et al. (2016), as Redes Neurais Convolucionais são simplesmente ANNs que usam convoluções no lugar de matrizes de multiplicação em pelo menos uma de suas camadas.

O que difere uma Rede Neural Convolucional de uma de outros tipos de ANN, como o Perceptron Multicamadas, é a sua estrutura de camadas, representada na Figura 4, que é composta por volumes de neurônios. Esta estrutura apresenta camadas espacialmente locais, ao menos em suas primeiras camadas. Uma camada em uma CNN recebe 3 dimensões de neurônios, com largura, altura e profundidade. Na camada de entrada, a altura e a largura são as dimensões da imagem e a profundidade é 3, para representar os canais RGB[3].

Diferentemente de uma ANN do tipo Perceptron Multicamadas, os neurônios de uma CNN não são do tipo totalmente conectados, ou seja, estão conectados apenas par-

[3]Abreviatura para o sistema de cores baseado nas cores vermelho (red), verde (green) e azul (blue).

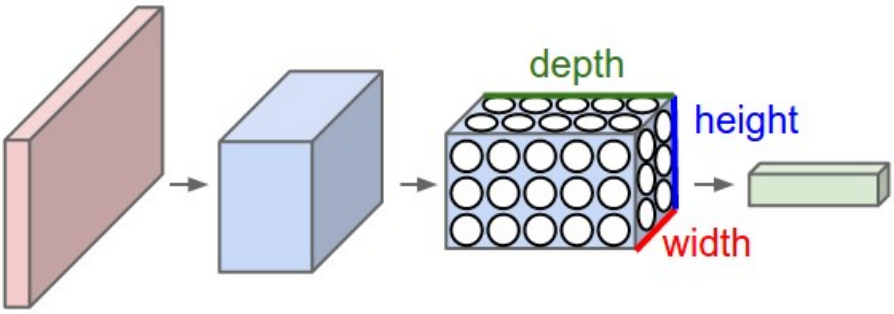


Figura 4. Exemplo de CNN, *Input Layer* a esquerda, ao longo de uma série de convoluções.

Fonte: Stanford (2023)

cialmente com a camada anterior. Apenas a camada de saída recebe essa estrutura totalmente conectada. Cada camada de uma CNN ativa um volume de neurônios, há também diversos tipos de camadas, sendo as mais comuns a camada convolucional, a camada de *pooling* e a camada totalmente conectada.

Uma convolução consiste em aplicar uma série de filtros, também conhecidos como *kernels*, em uma pequena região de *pixels* da imagem, ao longo de toda a sua extensão. O resultado de uma convolução é chamado de mapa de características. Nas palavras de Russell e Norvig (2022, p. 688), ”Um padrão de pesos que é replicado em várias regiões locais é chamado **kernel** e o processo de aplicação do kernel aos *pixels* da imagem é chamado convolução”. Na Figura 5, a convolução realizada aplica o produto escalar entre o filtro Sobel e a matriz delineada pela linha azul.

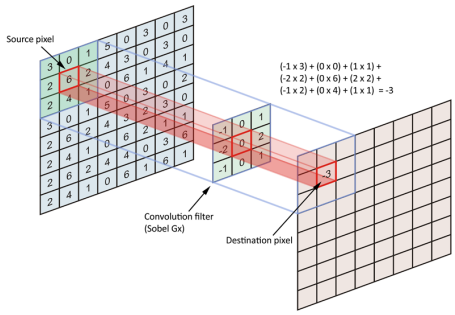


Figura 5. Operação de convolução do filtro Sobel.
Fonte : Lima et al. (2020)

O *Kernel* possui valores iguais para as dimensões de largura e altura, e também possui um *stride*, que define em quantos *pixels* o *kernel* deve deslocar o seu filtro. No *kernel* diferentes tipos de operações podem ser realizados, a depender das necessidades da aplicação. Uma operação muito comum utilizada em camadas convolucionais é a operação de *pooling*.

Uma camada de *pooling* de uma CNN busca diminuir as dimensões do volume e, consequentemente, reduzir a quantidade de parâmetros do modelo. No *pooling*, o *kernel* agrega os valores contidos em seu filtro. Embora existam diversos tipos de filtros, os

mais comuns são o *Average-pooling* - obtém o valor médio do filtro - e o *Max-pooling*, que obtém o valor máximo do filtro. Na Figura 6 é possivel vizualizar uma operação de *Max-pooling*.

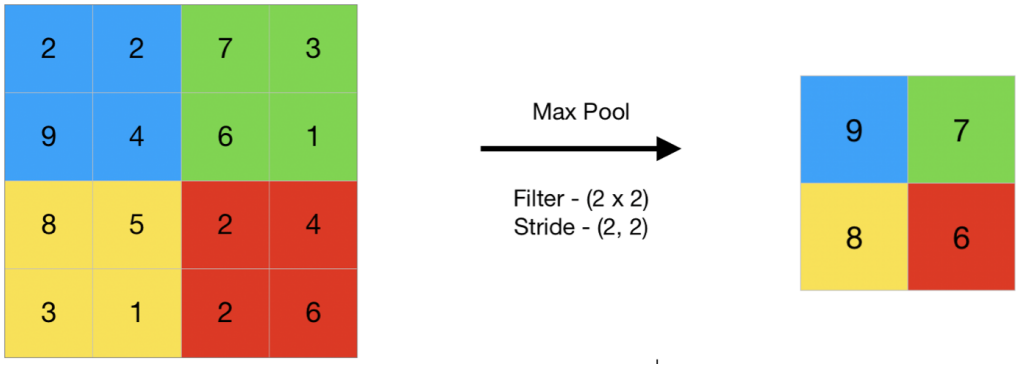


Figura 6. Max-Pooling aplicado em uma matriz 8x8, com um filtro 2x2 e stride 2x2.

Fonte : Wang (2018)

Na camada totalmente conectada, os neurônios aplicam transformações lineares nas entradas dos vetores, utilizando uma matriz de pesos. Após isso, uma transformação não linear é aplicada no produto através de uma função de ativação não linear. Geralmente, esta é a camada final e gera uma pontuação para cada uma das classes do volume.

2.3. Modelos

Nas subseções seguintes, os modelos utilizados nos experimentos são descritos em mais detalhes. São eles: *VGG16, VGG19, ResNet, ResNet50 e InceptionV3*. Todos os modelos citados foram participantes da competição *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)*[4], o que demonstra sua relevância, tendo em vista que a competição é bem conceituada e normalmente modelos que atingem o estado da arte estão presentes nela.

2.3.1. VGG

O *VGG* foi criado por Simonyan e Zisserman (2015). No trabalho, eles investigaram os efeitos que a profundidade de uma camada de CNN causa em sua acurácia. Segundo eles, sua maior contribuição foi a observação de que há uma significativa melhoria ao se utilizar entre 16 e 19 camadas, com convoluções 3 x 3.

Nessa arquitetura, durante o treinamento, a imagem passa por diversas pilhas de camadas convolucionais. O tamanho do *stride* na convolução é fixo em um (1) e o *padding* é adicionado de forma que a resolução seja preservada após a convolução. Algumas dessas camadas são seguidas por camadas de *max-pooling*, idênticas às presentes na Figura 6. Após esta estrutura, o modelo passa por três camadas totalmente conectadas, as duas primeiras possuem 4096 neurônios cada e a terceira 1000. Por fim é realizada a classificação desses 1000 canais com uma operação *softmax*.

[4]https://www.image-net.org/

O Quadro 1 apresenta o resumo das diferentes versões do *VGG*, identificadas como convX-Y, sendo que X representa o tamanho do filtro (3x3 ou 1x1) e Y o número de neurônios na camada.

As letras de A a E representam as diferentes versões. Cada nova versão possui uma quantidade maior de canais, ou seja, de camadas convolucionais do modelo, com exceção das versões C e D que possuem a mesma quantidade de camadas. A única diferença entre essas duas versões é o fato de que a versão C utiliza filtros 1 x 1, além dos filtros 3 x 3.

A arquitetura *VGG* demonstrou ter bom resultados, obtendo primeiro e segundo lugar nas competições de localização e classificação do desafio *ImageNet* 2014. Sua desvantagem é o alto custo computacional e grande quantidade de pesos, variando de 133 milhões de parâmetros na versão A, para 144 milhões na versão E.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

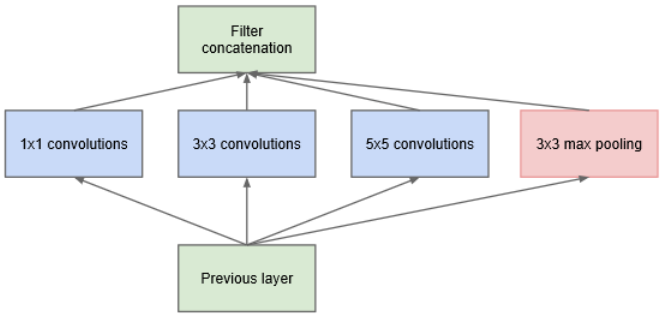
Quadro 1. Representadas pelas letras de A a E, estão as versões 11, 13, 16 e 19 do modelo VGG.

Fonte : Simonyan e Zisserman (2015)

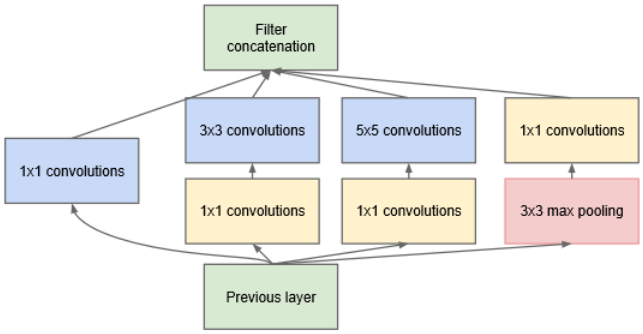
2.3.2. InceptionV3

Na tentativa de contornar os obstáculos gerados por um modelo muito grande, tais como maior propensão a *overfitting* e aumento de recursos computacionais, foi proposta uma nova arquitetura, chamada de *Inception* (Szegedy et al., 2014). Segundo Arora et al. (2013) apud Szegedy et al. (2014), uma rede neural profunda, larga e com conexões bastante esparsas pode ser otimizada camada por camada através de uma análise, observando a correlação dos neurônios com as saídas do modelo e realizando o agrupamento de neurônios nos casos onde a correlação é alta.

Essa otimização trás alguns efeitos colaterais no modelo. Szegedy et al. (2014) inferem que, nas primeiras camadas, neurônios com uma alta correlação estão concentrados em regiões isoladas da imagem e, como resultado, são criados muitos agrupamentos para uma mesma região. Isso torna mais fácil aplicar um filtro que cubra todos os neurônios responsáveis por aquela região, reduzindo o tamanho do *kernel* da operação de convolução. Por outro lado, neurônios em regiões locais esparsas agora estarão ainda mais afastados, sendo necessário um *kernel* maior. Por essas razões, são utilizadas convoluções de tamanhos 1x1, 3x3 e 5x5, juntamente com uma camada de *max-pooling*, adicionada paralelamente às convoluções. A Figura 5(a) demonstra como esse conjunto forma um módulo *Inception*:



(a) Inception module, naïve version



(b) Inception module with dimension reductions

Figura 5. Modulo *Inception*, sem a redução (a) e com a redução (b)
Fonte : Adaptado de Szegedy et al. (2014)

Conforme o modelo avança nas camadas mais profundas, ele atinge um nível

maior de abstração, diminuindo a concentração espacial, por conseguinte o número de operações 3x3 e 5x5 tende a aumentar e o de 1x1 a diminuir. Essas operações maiores podem ser inviáveis dependendo do contexto, pois este fator somado às camadas de *pooling* adicionadas paralelamente resultam em um aumento nas saídas de cada camada e consequentemente em um estouro de memória. A solução encontrada foi usar operações 1x1 antes das convoluções e após o *pooling*, como mostra a Figura 5(b).

De forma simplificada, a arquitetura é composta por módulos do tipo *a* e *b* presentes na Figura 5, intercalada com eventuais camadas de *pooling* para redução de dimensionalidade. O *InceptionV3* apresenta uma série de melhorias realizadas nessa estrutura de módulos *Inception*.

A primeira versão do *Inception* participou do desafio *ImageNet* 2014, ficando em primeiro lugar no teste de classificação a frente do *VGG*, que ficou em segundo lugar. Com as alterações propostas em Szegedy et al. (2015), o *InceptionV3* conseguiu obter melhores resultados que o *Inception* e o *VGG*, com um menor número de parâmetros do que o *VGG*.

2.3.3. ResNet

A arquitetura *ResNet* foi proposta por He et al. (2015) e significa *Residual Network*, pois seu modelo traz blocos residuais que visam resolver o problema de perca de desempenho ao longo do treinamento a medida que a rede fica mais profunda.

Uma das versões desta arquitetura implementa uma rede simples de 34 camadas e é baseada no modelo *VGG19*. Além disso, trás conexões diretas, que são chamadas de conexões residuais. A Figura 6 mostra um bloco residual. Para uma determinada camada, os valores de entrada *x* são adicionados ao final do bloco, após duas camadas de neurônios, antes da segunda operação ReLU. Esses atalhos ajudam no problema de degradação do gradiente - observado pelos autores - que faz com que o erro de treinamento aumente conforme a profundidade do modelo é aumentada.

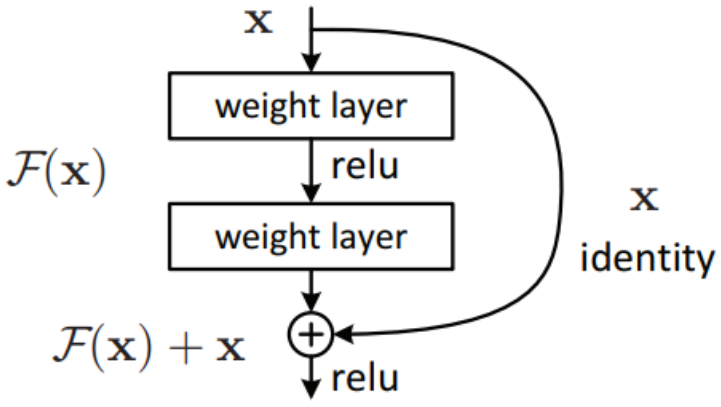


Figura 6. Funcionamento da aprendizagem residual
Fonte : He et al. (2015)

Este modelo possui camadas convolucionais 3x3 que seguem duas regras: 1) é que para os mapas de características com tamanhos iguais suas camadas também possuem a mesma quantidade de filtros e 2) se o mapa de características tiver metade do tamanho, o número de filtros são dobrados, visando manter a complexidade por camada. Ela também possui uma camada de *average pooling* global e uma camada totalmente conectada do tipo *1000-away* com função *softmax*.

No desafio *ImageNet*, a arquitetura *ResNet* obteve excelentes resultados e alcançou uma taxa de erro bastante pequena, de 4,49% nos resultados de modelo único e 3,57% com *ensembles*. Estes resultados foram menores do que arquiteturas como *VGG*, *GoogleLeNet*, entre outras (He et al., 2015). É possível visualizar as pontuações nas tabelas 1 e 2.

method	top-1 error	top-5 error
VGG (ILSVRC'14)	-	8.43
GoogleLeNet (ILSVRC'14)	-	7.89
VGG (v5)	24.4	7.1
PReLU-net	21.59	5.71
BN-inception	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Tabela 1. Taxas de erro em modelos únicos nos testes do ImageNet
Fonte : Figura adaptada de He et al. (2015)

method	top-5 error (test)
VGG (ILSVRC'14)	7.32
GoogleLeNet (ILSVRC'14)	6.66
VGG (v5)	6.8
PReLU-net	4.94
BN-inception	4.82
ResNet (ILSVRC'15)	**3.57**

Tabela 2. Taxas de erro em modelos ensembles nos testes do ImageNet
Fonte : Figura adaptada de He et al. (2015)

2.4. Data Augmentation

Para conseguir um desempenho satisfatório, as redes convolucionais profundas necessitam de uma grande quantidade de dados para serem treinadas de forma a evitar o *overfitting*. As técnicas de *Data Augmentation* (DA) visam aumentar o tamanho e melhorar a qualidade do *dataset*, permitindo que melhores modelos possam ser criados, com uma maior capacidade de generalização. São essenciais em situações em que não existe a possibilidade de obter mais imagens para o treinamento do modelo ou quando esta obtenção é muito cara.

As técnicas de *Data Augmentation* (Ampliação de Dados) aumentam o conjunto de dados artificialmente, deformando os dados *(data warping)* ou aumentando as amostras a partir dos dados originais (sobreamostragem ou *data oversampling*) (Shorten e Khoshgoftaar, 2019). As operações de *warp* transformam os dados de forma que sua classificação seja preservada, abrangendo transformações como as geométricas e as de cor. O uso de *oversampling* inclui técnicas como mesclagem de imagens e o uso de Redes Adversárias Generativas para criar novas imagens.

A necessidade do uso de DA varia de acordo com o domínio do problema e do conjunto de dados em questão. Em um *dataset* com grande parte das imagens tiradas a partir do mesmo ângulo, por exemplo, leva vantagem o uso de técnicas de translação, já em um *dataset* com imagens de mesma iluminação é mais proveitoso o uso de técnicas de manipulação de cores.

Segundo Shorten e Khoshgoftaar (2019) as técnicas de DA são similares a imaginar ou sonhar. Pessoas podem imaginar diferentes cenários baseados em sua experiência a fim de melhorar o entendimento sobre determinado assunto presente no nosso mundo. Desta forma as técnicas de DA também podem 'imaginar' alterações nas imagens para que deste modo possam gerar uma maior compreensão acerca delas e de suas características, possibilitando uma maior generalização para o modelo.

Quando se utiliza técnicas de DA é preciso ter cautela para preservar o rótulo correto do dado. Em um modelo que classifique números, rotacionar ou espelhar horizontalmente a imagem pode causar problemas na classificação entre o número 6 e o 9, por exemplo.

As técnicas de DA selecionadas neste trabalho são do tipo *warping* e são apresentadas a seguir.

2.4.1. Corte (Crop)

O corte na imagem pode ser feito de diversas maneiras, seja considerando uma região específica, como a parte central de uma imagem, eliminando apenas as bordas da imagem ou utilizando uma região aleatória. Também é possível usar um tamanho de corte específico ou um tamanho de corte aleatório. Nos casos em que a imagem necessita de um tamanho maior que o corte final é aplicada uma função de reescala, resultando em uma imagem com zoom e consequentemente, em uma imagem com uma resolução menor.

As técnicas de *crop* podem ajudar a CNN a generalizar melhor, evitando que a redes se adaptem demais a recursos ou características especificas que possam aparecer nas imagens (Takahashi et al., 2020).

Esse tipo de processamento pode alterar a classificação da imagem. Se no treinamento de modelos que reconhecem determinado tipo de animal, em uma imagem rotulada como positiva, for cortada a parte onde o animal aparece, a imagem pode ser classificada erroneamente e gerar uma inconsistência com seu rótulo.

Na Figura 7 é possível visualizar um exemplo de corte onde parte da girafa é cortada, dependendo do objetivo do modelo isso pode caracterizar uma perda de contexto, impedindo ou dificultando a classificação da imagem.

No exemplo, uma parte significativa do animal foi perdido por causa do corte. Essa restrição pode resultar em um modelo mais robusto, obrigando-o a classificar a imagem com uma região menor, mas também impedir uma classificação correta.

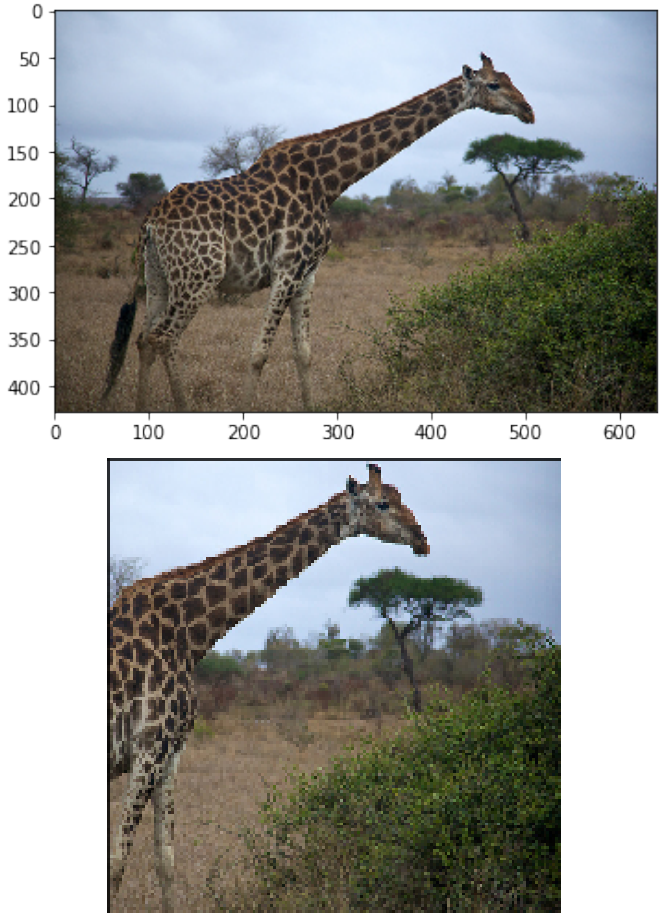


Figura 7. Exemplo de Corte, figura original acima.
Fonte : mxnet (2023)

2.4.2. Filtro de Kernel

O filtro de kernel funciona por meio do deslizamento de uma matriz n x n - conforme exemplificado no *max-pooling* na seção 2.2 - de um filtro de *blur* que segue uma distribuição Gaussiana, resultando em uma imagem borrada, ou um filtro de alto contraste em bordas horizontais ou verticais, resultando em imagens mais nítidas nas bordas (Shorten e Khoshgoftaar, 2019).

O uso da técnica de *blur* torna o modelo menos volátil a alterações na qualidade da imagem, como o *blur* de movimento e o desfoque. O *blur* de movimento é causado pelo tempo de exposição da iris da câmera, ou por objetos em movimento durante a captura da imagem (Vegas Creative Software, 2023). Dentre os tipos de *blur* destacam-se os que são utilizados neste trabalho, sendo eles *Gaussian*, *Median* e *Motion*, que são a convolução de uma função gaussiana na imagem, substituição de *pixels* aleatórios por uma mediana dos *pixels* adjacentes e desfoque de movimento, simulando um desfoque do movimento

da câmera, respectivamente.

Na Figura 8, é possível ver uma folha de uma planta à esquerda e um exemplo de *blur* à direita.

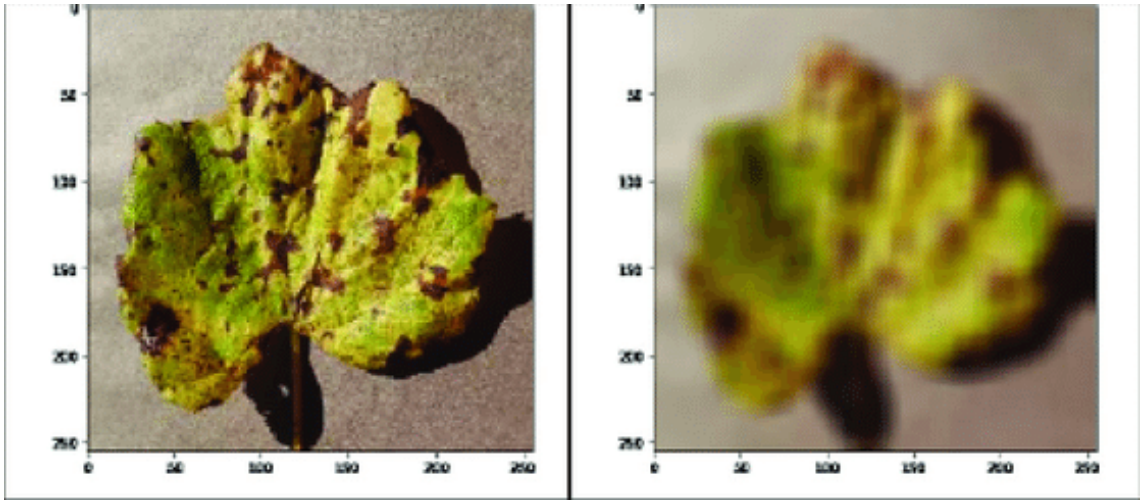


Figura 8. Exemplo de blur.
Fonte :Adaptada de Khan Ph.D. et al. (2020)

2.4.3. Transformações geométricas

As transformações geométricas alteram as imagens girando-as ao longo do eixo x ou eixo y, através de rotações, deformações geométricas e translações, por exemplo. Ao se utilizar essas transformações a distorção da imagem costuma ser mantida em um nível seguro para o modelo, dependendo do contexto das mesmas (Haba, 2023).

Por exemplo, em problemas para classificação de animais, como gatos ou cachorros, este tipo de transformação costuma não trazer riscos e tende a preservar as características que rotulam a imagem afetada pela técnica de *DA*. Por outro lado, em problemas como a classificação de dígitos numéricos, essas transformações podem impossibilitar a classificação correta. Um exemplo disto é a rotação dos números 6 e 9 citado na Seção 2.4 (Shorten e Khoshgoftaar, 2019).

As transformações geométricas modificam as imagens através da aplicação de operações matemáticas visando alterar a forma, escala e orientação da imagem e podem ser obtidas através operações como matrizes de transformação e vetores, por exemplo. Alguns exemplos dessa técnica são o espelhamento horizontal e o espelhamento vertical, utilizados neste trabalho, que aplicam operações que invertem os *pixels* da imagem, rotação, que gira a imagem com base no seu ponto central e redimensionamento.

2.4.4. Transformações de Cor

As transformações de cor envolvem aplicar alterações nos valores dos canais RGB dos *pixels* e na transparência (canal alfa) da imagem. É possível alterar as cores para tons mais escuros e obter uma diminuição no brilho da imagem. Essas técnicas podem ou

não preservar a rotulação da imagem, caso a cor seja uma informação importante. Elas são úteis para eliminar vieses de cor no *dataset*, em proveito de características espaciais (Shorten e Khoshgoftaar, 2019).

A Figura 9 apresenta diferentes filtros de cores.

Figura 9. Exemplos de transformações nos canais RGB.
Fonte : Retirado de mxnet (2023)

2.5. Trabalhos similares

Os trabalhos apresentados foram obtidos através de uma pesquisa no Google Scholar, utilizando a seguinte string de pesquisa: *("Convolutional Neural Networks"OR "CNN") AND ("data augmentation"OR "image augmentation") AND ("plant disease detection"OR "plant disease diagnosis")* publicados entre 2018 e 2023. Foram selecionados os 30 primeiros resultados obtidos pela pesquisa e, em seguida, foi realizada uma análise a fim de identificar se os trabalhos selecionados realmente utilizavam técnicas de *Data Augmentation* ou se apenas citavam seu uso na seção de trabalhos similares. Após essa etapa, os *abstracts* dos trabalhos restantes foram lidos e 10 deles foram selecionados para serem lidos completamente. Dos trabalhos lidos os mais relevantes são apresentados a seguir.

Negi (2021) realizou um estudo para analisar o desempenho de modelos de CNNs equipados com técnicas de DA, comparando-os com modelos que não utilizavam essas técnicas. Para a pesquisa, foi utilizada a base de imagens *PlantVillage*, que contém milhares de fotos de plantas infectadas por diferentes tipos de doenças. Durante a fase de treinamento, quatro técnicas de *Data Augmentation* foram aplicadas com o objetivo de gerar novas imagens: rotação, corte, escala (redução aleatória da imagem) e espelhamento (rotação aleatória da imagem).

Os resultados apontaram que as técnicas de DA podem melhorar o desempenho das CNNs na detecção de doenças em plantas. Além disso, foi destacado que para a classificação das folhas de tomate, todas as técnicas de *DA* resultaram em uma maior acurácia, Precisão, *Recall*, *F1-score* e *AUC-ROC* quando comparadas aos modelos sem DA. Dentre as técnicas utilizadas, o corte se destacou como a que obteve os melhores resultados, alcançando 88% de acurácia, 87% de Precisão, 89% de *Recall*, 88% de *F1-score* e 94% de *AUC-ROC*, como mostra a Tabela 3.

Para a classificação das folhas de batata também foi observado um aumento de desempenho em todas as métricas quando comparado ao modelo sem *DA*. Assim como

na folha de tomate, a técnica de corte se mostrou a mais eficaz, atingindo 91% de acurácia, 90% de Precisão, 92% de *Recall*, 91% de *F1-score* e 96% de *AUC-ROC*, como mostra a Tabela 4.

Modelo	Técnica de Data Augmentation	Acurácia	Precisão	Recall	F1-Score	AUC-ROC
CNN	Nenhum	0.85	0.84	0.86	0.85	0.92
CNN	Rotation	0.88	0.87	0.89	0.88	0.94
CNN	Scaling	0.86	0.85	0.87	0.86	0.93
CNN	Flipping	0.87	0.86	0.88	0.87	0.93
CNN	Cropping	0.88	0.87	0.89	0.88	0.94

Tabela 3. Classificação para folhas de tomate
Fonte : Adaptada de Negi (2021)

Modelo	Técnica de Data Augmentation	Acurácia	Precisão	Recall	F1-Score	AUC-ROC
CNN	Nenhum	0.87	0.86	0.88	0.87	0.93
CNN	Rotation	0.9	0.89	0.91	0.9	0.95
CNN	Scaling	0.88	0.87	0.89	0.88	0.94
CNN	Flipping	0.89	0.88	0.9	0.89	0.94
CNN	Cropping	0.91	0.9	0.92	0.91	0.96

Tabela 4. Classificação para folhas de batata
Fonte : Adaptada de Negi (2021)

Liang et al. (2019) propuseram uma nova rede neural baseada na arquitetura do *ResNet50* para o diagnóstico de doenças e classificação de severidade chamada PD²SE-Net. Esta rede contém estruturas residuais e unidades de embaralhamento. Além do modelo desenvolvido, também foram utilizadas técnicas de *DA* e visualização das redes neurais com o intuito de acelerar a escolha dos melhores hiperparâmetros durante o treinamento.

A base de dados utilizada nos experimentos realizados pelos autores continha nove espécies de plantas, cada uma com vários tipos de doenças e de duas categorias de severidade para cada doença, com dados gerados de forma artificial. Os autores utilizaram duas formas de *DA*: rotações e reflexões horizontais nas imagens de treinamento; e a adição aleatória de ruído (*Noise Injection*) e ruído gaussiano (*Gaussian Noise*).

O modelo mostrou bons resultados nas métricas *loss* e acurácia em todas as épocas do treinamento, alcançando 99%, 98% e 91% na classificação das espécies das plantas, diagnóstico da doença e severidade, respectivamente. Além disso, os autores destacaram que a menor acurácia de classificação de planta foi em torno de 97%, o que é semelhante ou superior a outros modelos testados no mesmo problema que não usam técnicas de *DA*. A acurácia em todos os diagnósticos de plantas ficou superior a 90%, exceto em uma das classes.

Abbas et al. (2021) propuseram uma rede neural para a detecção de doenças em tomates, utilizando a *C-GAN (Conditional Generative Adversarial Network)* para gerar

imagens sintéticas das folhas de tomate. O modelo *DenseNet121* foi treinado com imagens reais e as imagens sintéticas geradas pela *C-GAN* para classificar as folhas de tomate em dez categorias, sendo nove de doenças e uma de folhas saudáveis.

O trabalho de Abbas et al. (2021) consistiu em duas etapas principais. Primeiro, eles geraram imagens sintéticas usando a *C-GAN* como técnica de *DA* para evitar *overfitting* na rede neural. A *C-GAN* possui duas redes adversárias: uma para gerar imagens com base em um conjunto real e outra para distinguir as imagens reais das imagens geradas artificialmente. Em seguida, o modelo pré-treinado *DenseNet121* foi treinado usando tanto as imagens reais do *PlantVillage* quanto as imagens sintéticas geradas pela *C-GAN*.

O modelo proposto pelos autores alcançou excelentes resultados nos experimentos. Foram realizados testes utilizando apenas a base de imagens do *PlantVillage* e outros com as imagens geradas pela *C-GAN*. O modelo *DenseNet121*, quando treinado com as imagens sintéticas, obteve um desempenho maior em comparação ao modelo que utilizou apenas as imagens reais. Os autores também destacaram que a melhoria no desempenho indica que o uso das imagens geradas pela *C-GAN* ajudou na prevenção do *overfitting*, além de contribuir para uma melhor generalização do modelo. Além disso, os autores compararam os resultados obtidos com outros modelos, mas o *DenseNet121* apresentou os melhores resultados, atingindo uma acurácia de 94.34% e 97.11% nos modelos sem e com as imagens sintéticas, respectivamente.

Na Tabela 5 é possível visualizar os resultados obtidos com os modelos com e sem as imagens geradas pela *C-GAN* para diferentes quantidades de classes de doenças.

Modelo	Acurácia	Precisão	Recall	F1-Score
5 Classes				
DenseNet121	98.16%	0.83	0.97	0.98
DenseNet121 + Synthetic images	99.51%	0.99	0.99	0.99
7 Classes				
DenseNet121	95.08%	0.94	0.94	0.94
DenseNet121 + Synthetic images	98.65%	0.98	0.99	0.98
10 Classes				
DenseNet121	94.34%	0.95	0.92	0.93
DenseNet121 + Synthetic images	97.11%	0.97	0.97	0.97

Tabela 5. Acurácia, Precisão, Recall e f1-score para o modelo com e sem data augmentation

Fonte : Adaptada de Abbas et al. (2021)

Sagar e Jacob (2021) realizaram um estudo para comparar cinco diferentes arquiteturas pré-treinadas de redes neurais convolucionais para detecção de 38 tipos de doenças em plantas. Além disso, também foram utilizadas técnicas de *DA* pelos autores. As arquiteturas avaliadas foram *MobileNet*, *ResNet50*, *InceptionV3*, *InceptionResNetV2* e *DenseNet169*.

As imagens utilizadas pelos autores foram coletadas da base do *PlantVillage* e foram escolhidas 13 espécies de culturas, incluindo maçã, mirtilo, cereja, uva, laranja, pêssego, pimenta, batata, framboesa, soja, abóbora, morango e tomate. Foram utilizadas técnicas como zoom, alteração de brilho, espelhamento, cisalhamento, cortes e rotações.

Além das técnicas de *DA*, também foram utilizadas técnicas para visualização das camadas da rede a fim de acompanhar a aprendizagem do modelo e realizar os ajustes de parâmetros mais rapidamente, e *dropout*, que é uma técnica de regularização para auxiliar na prevenção do *overfitting*.

Após finalizar o treinamento dos modelos, os autores apresentaram os resultados obtidos comparando acurácia, Precisão, *Recall* e *F1-score*. O modelo que obteve os melhores resultados foi o *ResNet50*, alcançando 98% de acurácia, 94% de Precisão, 94% de *Recall* e 94% de *F1-score*.

Na tabela 6 são comparados os resultados obtidos entre os modelos utilizados. Embora os autores não tenham comparado os resultados gerados pelos modelos com as técnicas de *DA* (Tabela 6) com os resultados obtidos pelos mesmos modelos sem o uso dessas técnicas, percebe-se que elas geraram bons resultados, pois todas as métricas de todos os modelos são superiores a 0.90.

Métricas de avaliação	InceptionV3	InceptionResNetV2	ResNet50	MobileNet	DenseNet169
Acurácia	0.97	0.97	0.98	0.97	0.97
Precisão	0.92	0.91	0.94	0.94	0.92
Recall	0.94	0.93	0.94	0.93	0.93
F1	0.93	0.92	0.94	0.93	0.93

Tabela 6. Comparação dos modelos pré-treinados
Fonte : Adaptada de Sagar e Jacob (2021)

Nesta seção foi realizada uma análise de trabalhos que utilizaram *CNNs* e técnicas de *DA*. Essa combinação foi empregada com o objetivo de aumentar a quantidade de dados e aprimorar o desempenho dos modelos. Ao utilizar tais recursos, os autores destes trabalhos visaram não apenas aumentar a capacidade de generalização dos modelos, mas também diminuir as chances de *overfitting*.

Por meio dessa análise, foi possível observar que o uso de técnicas de *DA* pode desempenhar um papel importante na superação de problemas causados pela escassez de dados, assim como acontece neste trabalho de conclusão de curso. O entendimento das técnicas adotadas pelos autores lidos destaca a relevância da abordagem utilizada neste trabalho.

É importante destacar que não é possível realizar uma comparação direta entre o atual trabalho e os trabalhos apresentados, por conta de variações consideráveis tanto na detecção da doença quanto nas imagens utilizadas para o treinamento dos modelos.

Enquanto os trabalhos citados utilizaram bases de dados com grandes quantidades de imagens para treinar seus modelos, este trabalho usou uma quantidade aproximadamente 18 vezes menor do que o menor *dataset* utilizado dentro os trabalhos apresenta-

dos. Além disso, as imagens que foram utilizadas no atual trabalho foram produzidas por agricultores e agrônomos em ambientes diferentes e utilizando diferentes dispositivos, enquanto que os demais trabalhos possuíam, além de uma maior quantidade de imagens, maior qualidade e padronização.

Além disso, este trabalho se concentra na detecção de doenças presentes nos troncos das macieiras, enquanto os demais trabalhos, em sua maior parte, focam em detectar doenças através das folhas, onde as variações de cores causadas pelas doenças são mais aparentes e, portanto, mais facilmente percebidas. Já neste trabalho, as diferenças de cor causadas no tronco são muito mais sutis, tornando mais difícil a detecção.

3. Materiais e Métodos

No que se refere à natureza deste trabalho, sua classificação é de pesquisa aplicada e a abordagem do problema é quantitativa. O objetivo é realizar uma pesquisa descritiva, cujo procedimento técnico é uma pesquisa bibliográfica e experimental.

O trabalho foi desenvolvido em três etapas. A partir das imagens obtidas pelos fitopatogistas da Epagri e das imagens enviadas à plataforma Cancontrol através do *upload* pelos agricultures, deu-se início à primeira etapa. Nesta etapa, um dos objetivos é realizar uma inspeção manual nas imagens, a fim de eliminar àquelas que não sejam adequadas. Também é feita a preparação das imagens, de forma que sua utilização permita o melhor desempenho possível da rede.

A segunda etapa consiste em treinar uma rede neural para que ela seja capaz de classificar corretamente quais imagens estão contaminadas com a presença do fungo *Neonectria ditissima*. A rede foi desenvolvida na linguagem de programação Python, com o uso da biblioteca *Pytorch*. Os dados estatísticos foram criados com a biblioteca *scikit-learn*, as métricas salvas são Precisão, acurácia, F-Score, curva ROC, Recall, erro e FBeta com valor β de 0,5. Este último é a métrica escolhida para avaliar a qualidade do modelo, pois um valor $\beta < 1$ dá maior peso para a Precisão do que para o Recall, nesse caso a Precisão tem o dobro da importância no cálculo da métrica. Esta abordagem é utilizada para reduzir as chances de uma árvore contaminada ser classificada como saudável e contaminar outras árvores.

Por fim, na terceira etapa são aplicadas algumas técnicas de ampliação dos dados e os modelos são treinados e avaliados novamente.

3.1. Obtenção e Preparação das Imagens

As imagens obtidas são as mesmas do trabalho de Mattos e Ribeiro (2022). São 135 imagens, sendo 58 delas obtidas junto aos fitopatologistas da Epagri e as 77 restantes retiradas da base de dados da plataforma Cancontrol.

Três imagens foram descartadas devido à natureza do problema. Como a doença se manifesta primeiramente em troncos e galhos, a maioria dos diagnósticos não são feitos analisando o fruto da árvore. Por este motivo, apenas uma pequena quantidade das imagens têm o fruto presente nelas. Do total de imagens, somente três delas possuem somente o fruto, sem a árvore. Para simplificar o modelo, essas três imagens não foram utilizadas.

A Tabela 7 mostra a relação entre a origem das imagens e sua classificação.

Origem	Positivas	Negativas	Total
Epagri	34	24	58
Cancontrol	26	48	74
Total	60	72	132

Tabela 7. Distribuição das imagens por origem.

A Tabela 8 mostra a distribuição das imagens. Elas estão divididas em 60% para treino, 20% para validação e os 20% restantes para teste.

Conjunto	Positivas	Negativas	Total
Treino	35	42	77
Validação	12	15	27
Teste	13	15	28
Total	60	72	132

Tabela 8. Distribuição das imagens considerando os conjuntos e as classificações das imagens.

As imagens de treino são usadas pelo modelo para calcular as derivadas e atualizar os pesos da rede. As imagens de validação são utilizadas para avaliar o modelo, calcular o erro e outras métricas que permitem inferir se o modelo está melhorando ou piorando seu desempenho. O objetivo é fornecer ao modelo imagens diferentes das utilizadas durante o treino para obter resultados mais próximos da realidade. Quando se utiliza as imagens de treino para essa finalidade, a chance dos resultados estarem sobre o efeito de *overfitting* aumenta. Os resultados obtidos na validação servem como base para escolher o melhor modelo dado as peculiaridades do problema. Após o modelo ser escolhido, os dados de teste são utilizados para analisar o desempenho de teste do modelo e para simular qual seria a assertividade do modelo caso fosse posto em produção.

3.2. Indução dos modelos de classificação

O treinamento da rede neural foi feito utilizando cinco modelos pré-treinados diferentes, disponíveis no *framework torchvision*(TorchVision, 2016). A escolha desse *framework*, se deve a sua maior flexibilidade na construção e manutenção de arquiteturas, se comparado a outros *frameworks*. Ele também conta com uma grande comunidade ativa, o que confere mais facilidade na busca por informações ou solução de problemas.

Os modelos são: *Resnet*, nas versões com 18 e 50 camadas; *VGG*, nas versões com 16 e 19 camadas; e *Inception* na versão v3. Os modelos pré-treinados atingem bons resultados em geral e são treinados para classificar diversos tipos de imagens diferentes. A partir desses modelos pré-treinados é dado início ao treinamento do modelo deste trabalho. O Algoritmo 1 apresenta, em linhas gerais, os passos realizados nesse treinamento.

No Algoritmo 1, todos os argumentos do programa estão carregados na variável *parametros*, antes do início do programa. O *tamanhoBatch*, na linha 2, define até quantas imagens do conjunto são carregadas e treinadas no método *treinar*, na linha 9, antes dos

pesos da rede serem retropropagados pelo *Otimizador* na linha 13. Para este trabalho foi utilizado um *batch* de tamanho oito. Cada *batch* é processado por vez - gerando as saídas - até que todas as imagens do conjunto de treinamento sejam usadas. A partir das saídas geradas é possível utilizar a regra da cadeia para atualizar os pesos de todos os neurônios. Uma época é definida pelo processo de atualização dos pesos da rede após cada *batch*, até que todas as imagens tenham sido processadas uma vez. Neste trabalho, foi utilizado um número máximo de 40 épocas para treinamento.

Algoritmo 1 Fluxo básico do treinamento.

Entrada: parametros

```
1▸ numEpocas ← parametros.numEpocas
2▸ tamanhoBatch ← parametros.tamanhoBatch
3▸ modelo ← carregarModelo(parametros.nome_modelo)
4▸ dataset ← conjuntoDados(parametros.caminho, tamanhoBatch)
5▸ otimizador ← Otimizador(modelo.pesos, parametros.lr, parametros.wd)
6▸ melhorResultado ← 0
7▸ tolerancia ← parametros.tolerancia
8▸ para i ← 0 até num_epocas faça
9▸     resultadoTreino ← treinar(modelo, dataset.treino)
10▸    resultadoValidacao ← validar(modelo, dataset.validacao)
11▸    estatisticasTreino ← calculaMetricas(resultadoTreino)
12▸    estatisticasValidacao ← calculaMetricas(resultadoValidacao)
13▸    modelo ← otimizador.otimizarModelo(modelo)
14▸    se estatisticasValidacao.fscore > melhorResultado então

15▸        melhorResultado ← estatisticasValidacao.fscore
16▸        salvarModelo(modelo)
17▸        tolerancia ← parametros.tolerancia
18▸    senão
19▸        tolerancia ← tolerancia − 1
20▸        se tolerancia = 0 então
21▸            fim para
22▸        fim se
23▸    fim se
24▸    salvarEstatisticas(estatisticasVreino)
25▸
26▸    salvarEstatisticas(estatisticasValidacao)
27▸ fim para
28▸
29▸ resultadoTeste ← validar(modelo, dataset.teste)
30▸ estatisticasTeste ← calculaTetricas(resultadoTeste)
31▸ salvarEstatisticas(estatisticasTeste)
```

Após o treinamento é dado início ao processo de validação (*validar* na linha 10). Na validação, os pesos não são atualizados e o cálculo de gradientes não é feito, as saídas são geradas apenas para inferência do modelo. O método *calculaMétricas*, utilizado nas

linhas 11, 12 e 13, gera as métricas mencionadas anteriormente. O valor mais alto na métrica escolhida como critério de avaliação é registrado na variável *melhorResultado*. Caso a pontuação obtida na época atual seja maior do que a encontrada em *melhorResultado*, o modelo é salvo e o melhor valor é atualizado. A *tolerancia* é um número inteiro e define um critério de parada, se o algoritmo não evoluir em sua predição durante *x* épocas consecutivas, o treinamento é abortado. O fluxo de atualizar o melhor valor, salvar o modelo e atualizar a tolerância pode ser observado no bloco *se* nas linhas 14 a 23.

Na primeira rodada de experimentos, nove combinações de parâmetros foram utilizadas em cada um dos cinco modelos, totalizando 45 experimentos. Para obter um grau de confiança maior, cada experimento foi executado dez vezes. Os resultados foram utilizados para analisar se os diferentes fatores de *learning rate* (LR), *weight decay* (WD) e arquiteturas influenciam no resultado final e se essa influência ocorre de forma isolada, ou se há interação entre os fatores. Os diferentes modelos e valores de LR e WD utilizados estão disponíveis na lista abaixo, sendo que os experimentos foram realizados de forma fatorial.

- Modelos: ResNet18, ResNet50, VGG16, VGG19 e InceptionV3;
- LR: $1e^{-4}$, $1e^{-5}$ e $5e^{-5}$;
- WD: $1e^{-7}$, $1e^{-8}$ e $1e^{-9}$.

3.3. Avaliação dos Modelos

A matriz de confusão, apresentada no Quadro 2, é uma forma de visualizar os resultados de um modelo em um problema de classificação. Ela mostra o desempenho do modelo em cada classe, mostrando quantas entradas foram classificadas corretamente e quantas foram classificadas incorretamente.

	Predito: SIM	**Predito: NÃO**
Correto: SIM	Verdadeiro Positivo (VP)	Falso Negativo (FN)
Correto: NÃO	Falso Positivo (FP)	Verdadeiro Negativo (VN)

Quadro 2. Exemplo de matriz de confusão para um problema de classificação binária.

As métricas de avaliação de um sistema de classificação são calculadas a partir da matriz de confusão. Para avaliar e selecionar os melhores modelos, diversas métricas são salvas durante o treinamento do modelo, tais como, acurácia, Precisão *Recall, fscore* e curva roc.

A Acurácia mede a relação entre todos os acertos com a amostra inteira. Ela é definida pela Equação

$$\text{Acurácia} = \frac{VP + VN}{VP + VN + FP + FN} \tag{5}$$

Na maioria dos casos, uma única métrica não é o suficiente para medir a qualidade de uma modelo. Os conjuntos de dados podem ser desbalanceados - o número de ocorrências é desproporcional entre as classes - ou o problema pode exigir uma maior importância para a classe negativa, por exemplo. A Precisão mede a porcentagem de acertos positivos, em relação a todos os resultados classificados como positivos. Esta medida é dada pela Equação

$$\text{Precisão} = \frac{VP}{VP + FP} \tag{6}$$

O *Recall* mede a taxa de acertos positivos, em relação a todos os dados positivos, o que inclui os dados incorretamente classificados como negativos. Esta medida é definida pela Equação

$$\textit{Recall} = \frac{VP}{VP + FN} \tag{7}$$

Normalmente as medidas de Precisão e *Recall* são antagônicas, dependendo do problema um modelo pode apresentar um baixo *Recall*, mas uma alta Precisão e vice-versa. A Precisão mostra o quanto um modelo acerta quando ele diz que uma entrada é positiva. O modelo pode ter uma alta taxa de acertos, mas fazer poucas predições. Em contrapartida, o *Recall* mostra quantas imagens ele deixou passar, isto é, quantos dados positivos foram classificados como negativos. Um modelo pode não deixar nenhum dado positivo para trás, simplesmente classificando todos como positivos, porém isso reduz a Precisão consideravelmente.

O *F-Score* é uma medida que leva em conta tanto a Precisão quanto o *Recall*.

$$\textit{F-Score} = \frac{(1 + \beta^2) * \text{Precisão} * \textit{Recall}}{\beta^2 * \text{Precisão} + \textit{Recall}} \tag{8}$$

O *F-Score* é uma média harmônica entre a Precisão e o *Recall*. O valor β pode ser utilizado com valores diferentes de 1 para dar maior ênfase para o *Recall* ou para a Precisão, sendo $\beta > 1$ mais ênfase para o *Recall* e $\beta < 1$ mais importância para a Precisão.

Neste trabalho é utilizado o valor de β igual a 0.5 para o *F-Beta* a fim de dar mais importância para a Precisão já que dado o contexto do problema abordado ela é mais relevante do que o *Recall*, uma vez que os casos rotulados como saudáveis serão analisadas por especialistas, com intuito de evitar os falsos negativos e permitir que uma árvore supostamente saudável contamine outras.

3.4. Implementação das Técnicas de Data Augmentation

Nesse experimento foram utilizadas as bibliotecas *albumentations* (Buslaev et al., 2020) e *torchvision* (TorchVision, 2016) para aplicar as técnicas escolhidas. Antes de realizar as transformações nas imagens com o *albumentations*, é necessário que elas estejam de acordo com as restrições da biblioteca. As imagens precisam ser carregadas de forma que as dimensões de sua matriz de *pixels* estejam no formato altura, largura e canais

e que estejam no formato de cores RGB no intervalo 0 a 255. Depois de realizar as transformações, as dimensões da matriz são convertidas para o formato canais, altura, largura, no formato RGB de 0 a 1. Essas alterações no formato das imagens se deve à diferença do formato usado pelas entre as bibliotecas *albumentations* e *pytorch*.

3.4.1. Corte

Para implementar as técnicas de corte descritas em 2.4.1 , são utilizadas as funções *CenterCrop*, *CropAndPad*, *RandomCrop* e *RandomResizedCrop*, presentes em ambas as bibliotecas. Neste trabalho, todos os cortes foram realizados utilizando as funções da biblioteca *torchvision*.

Nos cortes *CenterCrop* e *CropAndPad* é feito um corte central na imagem. Em alguns dos cortes utilizados (incluindo o *CenterCrop*), a imagem é reescalada do seu tamanho original para um tamanho cerca de 15% maior do que o tamanho final desejado. No contexto do *CenterCrop*, a ampliação em relação ao tamanho desejado permite que o resultado após a transformação esteja no tamanho correto, evitando a necessidade de redimensionar a imagem causando uma perda na qualidade. Em ambos os cortes mencionados, a largura do corte pode ser configurada em cada um dos quatro lados da imagem separadamente. A ausência de aleatoriedade torna esses dois cortes determinísticos. O *CropAndPad* é o único dos cortes selecionados que não amplia a imagem. Nele, ao eliminar as bordas de forma idêntica ao corte central, o espaço eliminado é preenchido com um fundo da cor preta.

No *RandomCrop* o tamanho do corte é especificado utilizando apenas dois parâmetros, a altura e a largura. Uma região aleatória é escolhida a cada iteração e o resultado produzido é diferente. A ampliação na imagem é utilizada pelos mesmos motivos apresentados no *CenterCrop*.

O *RandomResizedCrop* funciona de maneira análoga ao *RandomCrop*, sua diferença reside no tamanho do corte, que também é aleatório. Um intervalo entre 0 e 1 é especificado e o tamanho será um valor entre o limite inferior e superior. Pelo fato do tamanho ser variável, uma parte dos cortes terão o tamanho final menor do que o necessário para o treinamento. A ampliação antes do corte é feita para mitigar os efeitos do redimensionamento na imagem após o corte.

Também há uma restrição na área disponível para o corte ser aplicado, pois conforme mencionado na subseção *Crop* 2.4.1, é preciso cuidar para que não haja uma alteração no rótulo da imagem. Dessa forma, limitando a área disponível evitam-se regiões desfavoráveis, como cantos e bordas, que geralmente representam áreas menos relevantes para a categorização da imagem.

3.4.2. Filtro de Kernel (*Blur*)

Para implementar os filtros de kernel apresentados na seção 2.4.2 são utilizadas as funções *MedianBlur*, *MotionBlur* e *Gaussian*.

O *MotionBlur* simula um *blur* de movimento, produzindo um resultado similar

ao de tirar uma foto com a câmera em movimento. Pode ser simulado com o uso de um vetor de velocidade, para criar diferentes quadros da mesma foto, gerando uma média deles para aplicar o filtro de *blur* (Nvidia, 2023). No *MedianBlur* o filtro deslizante utiliza como critério a mediana dos valores do *kernel*. No *Gaussian* o efeito criado é similar ao de visualizar a imagem através de um objeto translúcido. Essa técnica é normalmente utilizada para reduzir o ruído e os detalhes da imagem.

3.4.3. Transformações geométricas

No grupo de transformações geométricas estão presentes diversos tipos de de transformações que permitem que se altere a geometria das imagens. Dentre essas técnicas foram selecionadas da biblioteca *Albumentations* para serem utilizas neste trabalho o espelhamento vertical (*vertical flip*) e o espelhamento horizontal (*horizontal flip*) que aplicam uma operação onde são invertidos os *pixels rgb* da matriz. No *horizontalFlip* os *pixels* mais à direita se transformam nos mais à esquerda e vice-versa, no *verticalFlip* os *pixels* do topo se transformam nos *pixels* inferiores.

3.4.4. Transformações de cor

Dentre as técnicas que estão presentes no grupo de transformações de cor foram escolhidas para serem utilizadas neste trabalho as técnicas *RandomBrightNessContrast* e *ColorJitter*, que modificam de forma randômica o brilho, contraste, saturação e propriedades de cor das imagens. O *RandomBrightNessContrast* realiza alterações no brilho e contraste das imagens e o *ColorJitter* faz transformações em propriedades de cor, brilho, contraste e saturação. A técnica de *ColorJitter* está presente na biblioteca *torchvision*. O *RandomBrightNessContrast* foi criado a partir do *ColorJitter*, mantendo a matiz de cores e a saturação inalterados.

4. Resultados

Conforme detalhado na seção 3, os resultados apresentados nessa seção foram obtidos através do treinamento dos cinco modelos de CNN selecionados *InceptionV3*, *VGG16*, *VGG19*, *ResNet* e *ResNet50*. Foram realizados dois tipos de experimentos, um sem o uso de técnicas de *Data Augmentation* e outro com o uso de algumas técnicas de *Data Augmentation*. As imagens utilizadas para ambos os treinamentos foram separadas em três partes distintas, uma para treinamento com 60% das imagens, outra para a validação com 20%, e outra para teste com os 20% restantes. Além disso, foram realizados testes com combinações diferentes dos hiperparâmetros *learning rate* e *weight decay* a fim de encontrar a configuração que gerasse os melhores resultados.

4.1. Modelos sem *Data Augmentation*

Nesta primeira etapa dos treinamentos foi realizada uma série de experimentos com diferentes valores de *learning rate* e *weight decay* com intuito de alcançar os melhores resultados sem o uso das técnicas de *Data Augmentation*. Para cada um dos cinco modelos selecionados foram experimentadas nove combinações diferentes de *learning rate* e *weight decay*. Além disso foram realizadas dez repetições para cada modelo em

cada configuração distinta, o que resultou em um total de 450 experimentos. As nove combinações de *learning rate* e *weight decay* escolhidas foram $1e^{-4}$, $1e^{-5}$ e $5e^{-5}$ para o *learning rate* e $1e^{-7}$, $1e^{-8}$ e $1e^{-9}$ para o *weight decay*.

Após a finalização de todos os experimentos foi realizada a análise de variância *ANOVA*, que pode ser visualizada na Tabela 9 com o objetivo de medir a influência de cada fator e suas interações nos resultados obtidos. Ao realizar a inspeção dos resultados da análise de variância *ANOVA* no conjunto de dados de validação, pode-se perceber que os diferentes modelos, diferentes valores de *learning rate* e as interações entre estes geraram uma influencia nos resultados, como é possível observar na coluna $PR(> F)$, os quais atingiram $9.0112318223e^{-24}$, $4.3166784722e^{-18}$ e $5.0261641571e^{-07}$ respectivamente, o que indica que tanto os fatores individuais quanto as interações entre eles foram significativas já que seus valores foram inferiores a 0.05. Por outro lado, tanto o fator individual do *weight decay*, quanto suas interações ficaram com valores acima de 0.05, o que indica que o *weight decay* não tem influência significativa nos resultados.

Factors	**df**	**sum_sq**	**mean_sq**	**F**	**PR(>F)**
C(model)	4.0	2.3793	0.5948	32.9081	9.0112e-24
C(lr)	2.0	1.5980	0.7990	44.2046	4.3166e-18
C(wd)	2.0	0.0540	0.0270	1.4964	2.2516e-01
C(model):C(lr)	8.0	0.8394	0.1049	5.8047	5.0261e-07
C(model):C(wd)	8.0	0.2736	0.0342	1.8921	5.9719e-02
C(lr):C(wd)	4.0	0.1345	0.0336	1.8611	1.1641e-01
C(model):C(lr):C(wd)	16.0	0.2470	0.0154	0.8543	6.2296e-01
Residual	405.0	7.3207	0.0180		

Tabela 9. Análise de variança Anova para modelos, *learning rate* e *weight decay* no conjunto de validação.

Por fim, foi gerado um gráfico de violino para observar a variação do *weight decay* para os valores de *F-Beta*, que pode ser visualizado na Figura 10, com intuito de selecionar apenas um dos valores testados para o *weight decay*, já que seus valores não apresentaram alterações significativas nos resultados. Ao analisar o gráfico pode-se notar que o valor $1e^{-7}$ de *weight decay* conferiu uma maior estabilidade aos resultados, além disso este valor também gerou menos resultados mais baixos quando comparado com os demais valores de *weight decay*. Por isso o valor de $1e^{-7}$ foi escolhido como parâmetro fixo para o *weight decay* para ser utilizado nos experimentos posteriores com *Data Augmentation*.

A Tabela 10 apresenta a média dos resultados do teste com diferentes métricas dos experimentos realizados nesta primeira etapa sem a utilização das técnicas de *Data Augmentation*. Nesta tabela, pode-se observar qual valor de *learning rate* levou cada modelo ao melhor resultado *F-Beta*, que foi considerada a métrica mais importante no contexto deste trabalho. São eles:

- InceptionV3 com *learning rate* de 1e-4 com F-Beta de 57,18%;
- ResNet50 com *learning rate* de 1e-4 com F-Beta de 72,24%;
- VGG16 com *learning rate* de 1e-4 com F-Beta de 76,29%;

- ResNet com *learning rate* de 5e-5 com F-Beta de 65,26%;
- VGG19 com *learning rate* de 1e-4 com F-Beta de 79,55%.

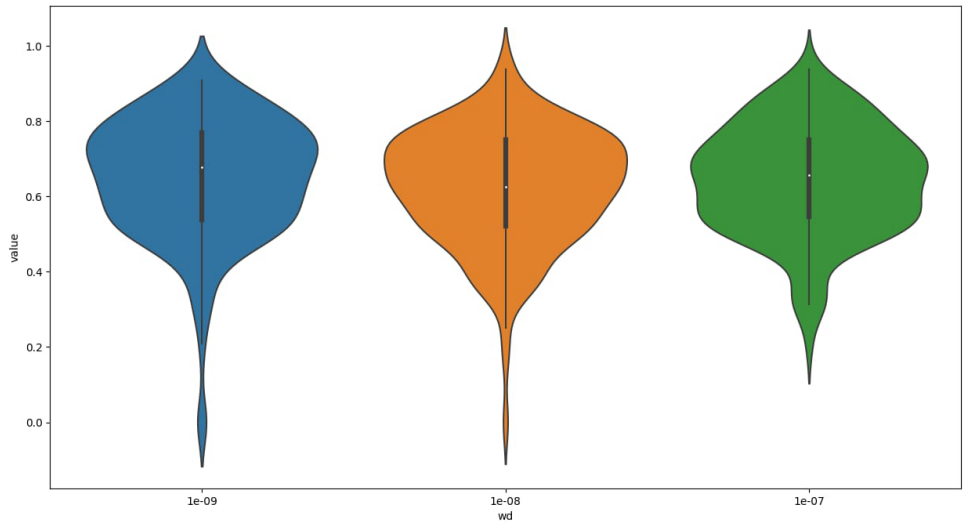


Figura 10. Gráfico de violino para *weight decay* com valores de *F-Beta* do conjunto de validação

Modelo	LR	Acurácia	Precisão	Recall	F-Beta	AUC
InceptionV3	$1e^{-4}$	0.6185	0.7498	0.4583	0.5718	0.625
InceptionV3	$1e^{-5}$	0.6037	0.6169	0.5333	0.5427	0.55
InceptionV3	$5e^{-5}$	0.5666	0.6265	0.45	0.4932	0.5244
ResNet50	$1e^{-4}$	0.7259	0.7820	0.6	0.7224	0.7294
ResNet50	$1e^{-5}$	0.6370	0.6223	0.5	0.5635	0.6388
ResNet50	$5e^{-5}$	0.6259	0.6984	0.4166	0.5859	0.5755
VGG16	$1e^{-4}$	0.7592	0.8356	0.5833	0.7629	0.7605
VGG16	$1e^{-5}$	0.6518	0.6606	0.725	0.6299	0.6127
VGG16	$5e^{-5}$	0.7296	0.8718	0.5583	0.7569	0.7088
ResNet	$1e^{-4}$	0.6259	0.6019	0.6000	0.5917	0.6244
ResNet	$1e^{-5}$	0.6333	0.6732	0.6416	0.6205	0.6494
ResNet	$5e^{-5}$	0.6814	0.7003	0.575	0.6526	0.6427
VGG19	$1e^{-4}$	0.7777	0.8788	0.5916	0.7955	0.8261
VGG19	$1e^{-5}$	0.6518	0.7020	0.5	0.5924	0.6666
VGG19	$5e^{-5}$	0.7407	0.8873	0.5083	0.7606	0.7688

Tabela 10. Resultados dos modelos sem técnicas de *Data Augmentation* no conjunto de validação.

Os dois modelos com melhores resultados em seu F-Beta médio, foram o VGG16 e o VGG19. Em ambos os casos, a configuração com *learning rate* $5e^{-5}$ trouxe melhores

resultados na Precisão média, porém o *learning rate* $1e^{-4}$ aumentou o Recall médio, o que fez com que o F-Beta médio dos modelos aumentasse nessa configuração. O melhor modelo, VGG19, atingiu um F-Beta de 0.7955, enquanto que o VGG16 atingiu um F-Beta de 0.7629, ambos com *learning rate* $1e^{-4}$.

De todas as combinações testadas, os experimentos com DA foram feitos apenas com os *learning rates* que obtiveram o melhor F-Beta para cada modelo. As combinações podem ser visualizadas na Tabela 11, na qual são mostrados os menores e o maiores valores de F-Beta atingidos em cada modelo, bem como a média do F-Beta nas 10 replicações.

Modelo	LR	F-Beta(Min)	F-Beta(Média)	F-Beta(Max)
InceptionV3	$1e^{-4}$	0.3125	0.5718	0.7291
ResNet50	$1e^{-4}$	0.5208	0.7224	0.875
VGG16	$1e^{-4}$	0.6730	0.7629	0.9090
ResNet	$5e^{-5}$	0.5357	0.6526	0.8653
VGG19	$1e^{-4}$	0.6730	0.7955	0.875

Tabela 11. Resultados dos modelos sem técnicas de *Data Augmentation* no conjunto de validação.

4.2. Modelos com *Data Augmentation*

Nesta segunda etapa dos experimentos foram realizados testes utilizando diversas técnicas de *Data Augmentation*, que foram explicadas na seção 3.4, com intuito de atingir melhores resultados dos que foram obtidos na primeira etapa citada na seção 4.1.

Dos 5 modelos utilizados, o *learning rate* $1e^{-4}$ é utilizado nos modelos ResNet50, VGG16, VGG19 e InceptionV3, o ResNet18 é o único a utilizar o valor $5e^{-5}$. Diversos experimentos individuais foram realizados, cada técnica foi aplicada independente da outra sem a combinação das técnicas. Também foi feito um experimento combinando uma técnica de cada categoria, com o objetivo de verificar se as técnicas em conjunto levam a melhores resultados do que separadamente. Em ambos os casos foram feitas 10 replicações e os resultados são apresentados nas próximas subseções, todos utilizando como referência o conjunto de dados de validação.

4.2.1. Corte

Os resultados da aplicação da técnica de *CenterCrop* estão disponíveis na Tabela 12. Esta técnica não apresentou resultados positivos quando utilizada de forma individual. Todos os valores de F-Beta médio e máximo foram inferiores, se comparados aos resultados sem DA.

Modelo	F-Beta(Min)		F-Beta(Média)		F-Beta(Max)	
	Sem DA	DA	Sem DA	DA	Sem DA	DA
InceptionV3	0.3125	0.3125	0.5718	0.5280	0.7291	0.75
ResNet50	0.5208	0.5555	0.7224	0.7218	0.875	0.8333
VGG16	0.6730	0.625	0.7629	0.7491	0.9090	0.875
ResNet	0.5357	0.2884	0.6526	0.4933	0.8653	0.5882
VGG19	0.6730	0.4687	0.7955	0.7297	0.875	0.9090

Tabela 12. Resultados dos modelos usando a técnica de *CenterCrop* no conjunto de validação.

Os resultados da técnica de *CropAndPad* estão disponíveis na Tabela 13. Todos os modelos, com exceção do ResNet50, tiveram redução nas 3 métricas. No modelo ResNet50 o F-Beta mínimo aumentou de 0.5208 para 0.6818, o F-Beta médio aumentou de 0.7224 para 0.7607 e o F-Beta máximo se manteve constante.

Modelo	F-Beta(Min)		F-Beta(Média)		F-Beta(Max)	
	Sem DA	DA	Sem DA	DA	Sem DA	DA
InceptionV3	0.3125	0.2631	0.5718	0.5551	0.7291	0.6944
ResNet50	0.5208	0.6818	0.7224	0.7607	0.875	0.875
VGG16	0.6730	0.4166	0.7629	0.6814	0.9090	0.8333
ResNet	0.5357	0.4411	0.6526	0.5998	0.8653	0.8333
VGG19	0.6730	0.5555	0.7955	0.6914	0.875	0.8333

Tabela 13. Resultados dos modelos usando a técnica de *CropAndPad* no conjunto de validação.

Os resultados da aplicação da técnica de *RandomCrop* estão disponíveis na Tabela 14. O único modelo com melhorias em seu F-Beta médio foi o InceptionV3. Os seus valores de mínimo, média e máxima passaram de 0.3125, 0.5718 e 0.7291 para 0.4166, 0.6405 e 0.7954 respectivamente.

Modelo	F-Beta(Min)		F-Beta(Média)		F-Beta(Max)	
	Sem DA	DA	Sem DA	DA	Sem DA	DA
InceptionV3	0.3125	0.4166	0.5718	0.6405	0.7291	0.7954
ResNet50	0.5208	0.625	0.7224	0.7260	0.875	0.8333
VGG16	0.6730	0.625	0.7629	0.7237	0.9090	0.8333
ResNet	0.5357	0.125	0.6526	0.4867	0.8653	0.7291
VGG19	0.6730	0.5	0.7955	0.7907	0.875	0.9090

Tabela 14. Resultados dos modelos usando a técnica de *RandomCrop* no conjunto de validação.

Os resultados da técnica *RandomResizedCrop* estão disponíveis na Tabela15. Os

modelos InceptionV3 e ResNet50 tiveram melhorias em seus valores médios, de 0.5718 para 0.5975 no InceptionV3 e de 0.7224 para 0.7321 no ResNet50.

Modelo	F-Beta(Min)		F-Beta(Média)		F-Beta(Max)	
	Sem DA	DA	Sem DA	DA	Sem DA	DA
InceptionV3	0.3125	0.4166	0.5718	0.5975	0.7291	0.75
ResNet50	0.5208	0.5357	0.7224	0.7321	0.875	0.875
VGG16	0.6730	0.625	0.7629	0.7439	0.9090	0.8333
ResNet	0.5357	0.375	0.6526	0.6297	0.8653	0.7692
VGG19	0.6730	0.625	0.7955	0.7680	0.875	0.9090

Tabela 15. Resultados dos modelos usando a técnica de *RandomResizedCrop* no conjunto de validação.

4.2.2. Blur

Os resultados da técnica *Median* estão disponíveis na Tabela 16. Todos os modelos testados foram inferiores aos sem DA, em suas 3 métricas.

Modelo	F- Beta(Min)		F-Beta(Média)		F-Beta(Max)	
	Sem DA	DA	Sem DA	DA	Sem DA	DA
InceptionV3	0.3125	0	0.5718	0.3615	0.7291	0.5769
ResNet50	0.5208	0	0.7224	0.3875	0.875	0.625
VGG16	0.6730	0	0.7629	0.3031	0.9090	0.5172
ResNet	0.5357	0	0.6526	0.2535	0.8653	0.5357
VGG19	0.6730	0	0.7955	0.3513	0.875	0.7291

Tabela 16. Resultados dos modelos usando a técnica de *Median* no conjunto de validação.

Os resultados obtidos com o uso da técnica *Motion* estão disponíveis na Tabela 17. Assim como no uso do *Median*, os resultados foram inferiores em suas 3 métricas.

Modelo	F-Beta(Min)		F-Beta(Média)		F-Beta(Max)	
	Sem DA	DA	Sem DA	DA	Sem DA	DA
InceptionV3	0.3125	0	0.5718	0.4818	0.7291	0.6944
ResNet50	0.5208	0	0.7224	0.3535	0.875	0.5357
VGG16	0.6730	0	0.7629	0.3959	0.9090	0.5555
ResNet	0.5357	0	0.6526	0.4504	0.8653	0.5357
VGG19	0.6730	0	0.7955	0.4956	0.875	0.8333

Tabela 17. Resultados dos modelos usando a técnica de *Motion* no conjunto de validação.

Os resultados obtidos com a implementação da técnica *Gaussian* estão disponíveis na Tabela 18. Os únicos modelos que apresentaram melhorias foram o InceptionV3 e o VGG19, esses modelos tiveram seus valores de F-Beta médio aumentados de 0.5718 para 0.5870 e de 0.7955 para 0.8072 respectivamente.

Modelo	F-Beta(Min)		F-Beta(Média)		F-Beta(Max)	
	Sem DA	DA	Sem DA	DA	Sem DA	DA
InceptionV3	0.3125	0.4166	0.5718	0.5870	0.7291	0.7352
ResNet50	0.5208	0.5	0.7224	0.675	0.875	0.8333
VGG16	0.6730	0.625	0.7629	0.7417	0.9090	0.8333
ResNet	0.5357	0.3571	0.6526	0.6070	0.8653	0.8088
VGG19	0.6730	0.6944	0.7955	0.8072	0.875	0.9090

Tabela 18. Resultados dos modelos usando a técnica de *Gaussian* no conjunto de validação.

4.2.3. Transformações Geométricas

Os resultados do experimento com a técnica *Horizontal* estão disponíveis na Tabela 19. O InceptionV3 obteve um aumento de 0.04 em seu F-Beta médio e um aumento de 0.08 em seu F-Beta máximo. O VGG19 teve um aumento de 0.02 em seu valor médio e um aumento de 0.06 em seu valor máximo.

Modelo	F-Beta(Min)		F-Beta(Média)		F-Beta(Max)	
	Sem DA	DA	Sem DA	DA	Sem DA	DA
InceptionV3	0.3125	0.3571	0.5718	0.6144	0.7291	0.8035
ResNet50	0.5208	0.625	0.7224	0.7193	0.875	0.8333
VGG16	0.6730	0.625	0.7629	0.7291	0.9090	0.8333
ResNet	0.5357	0.5	0.6526	0.6095	0.8653	0.6944
VGG19	0.6730	0.6818	0.7955	0.8128	0.875	0.9375

Tabela 19. Resultados dos modelos usando a técnica de *Horizontal* no conjunto de validação.

Na Tabela 20 é possível visualizar os resultados ao aplicar a técnica *Vertical*. Nessa técnica deve ser destacado o modelo ResNet50 entre os demais. Foi o único modelo com melhorias em seu valor médio, que passou de 0.7224 para 0.7670, porém o valor máximo caiu de 0.875 para 0.8333.

Modelo	F-Beta(Min)		F-Beta(Média)		F-Beta(Max)	
	Sem DA	DA	Sem DA	DA	Sem DA	DA
InceptionV3	0.3125	0.375	0.5718	0.5544	0.7291	0.7142
ResNet50	0.5208	0.6944	0.7224	0.7670	0.875	0.8333
VGG16	0.6730	0.6818	0.7629	0.7543	0.9090	0.8333
ResNet	0.5357	0.3289	0.6526	0.5590	0.8653	0.7954
VGG19	0.6730	0.6944	0.7955	0.7950	0.875	0.9090

Tabela 20. Resultados dos modelos usando a técnica de *Vertical* no conjunto de validação.

4.2.4. Transformações de Cor

Os resultados da aplicação da técnica *Brightness* estão disponíveis na Tabela 21. O uso dessa técnica resultou em uma queda no desempenho de todos os modelos. O InceptionV3 perdeu 0.07 em seu valor médio, assim como o ResNet, que também teve uma redução de 0.07. Os outros modelos também tiveram seus valores de F-Beta médio reduzidos, embora de forma mais sutil.

Modelo	F-Beta(Min)		F-Beta(Média)		F-Beta(Max)	
	Sem DA	DA	Sem DA	DA	Sem DA	DA
InceptionV3	0.3125	0.3846	0.5718	0.5057	0.7291	0.6818
ResNet50	0.5208	0.5	0.7224	0.7137	0.875	0.875
VGG16	0.6730	0.625	0.7629	0.7120	0.9090	0.7692
ResNet	0.5357	0.4166	0.6526	0.5838	0.8653	0.75
VGG19	0.6730	0.6818	0.7955	0.7716	0.875	0.9375

Tabela 21. Resultados dos modelos usando a técnica de *Brightness* no conjunto de validação.

Os resultados do *ColorJitter* estão disponíveis na Tabela 22. Com exceção do VGG19 todos os modelos perderam desempenho, e tiveram seus valores de F-Beta médio e máximo reduzidos. O VGG19 aumentou seu valor médio de 0.7955 para 0.8003, enquanto seu valor máximo se manteve constante em 0.875.

Modelo	F-Beta(Min)		F-Beta(Média)		F-Beta(Max)	
	Sem DA	DA	Sem DA	DA	Sem DA	DA
InceptionV3	0.3125	0.3571	0.5718	0.5214	0.7291	0.7142
ResNet50	0.5208	0.5681	0.7224	0.6951	0.875	0.8333
VGG16	0.6730	0.6666	0.7629	0.7439	0.9090	0.875
ResNet	0.5357	0.5	0.6526	0.6211	0.8653	0.7812
VGG19	0.6730	0.6818	0.7955	0.8003	0.875	0.875

Tabela 22. Resultados dos modelos usando a técnica de *ColorJitter* no conjunto de validação.

4.2.5. Técnicas Combinadas

Nessa etapa dos experimentos, os modelos apresentados na seção 4.2 foram utilizados para realizar um experimento combinando diferentes técnicas, com as mesmas configurações de *learning rate*. Foi escolhida uma única técnica para cada categoria presente na seção 3.4.

No Corte a técnica de *CenterCrop* foi descartada por apresentar resultados inferiores na média de todos os modelos, o *CropAndPad* trouxe uma melhoria apenas no ResNet50, todos os outros modelos perderam desempenho, com destaque ao VGG19, que perdeu 0.1 em sua média. Por ser o modelo com o melhor desempenho sem DA (0.7955), foi dado uma maior prioridade ao modelo VGG19. Usando o mesmo critério o *RandomResizedCrop* foi excluído por reduzir o F-Beta médio do VGG19 de 0.7955 para 0.7680, o *RandomCrop* trouxe uma redução de apenas 0.0048 e dentre as 4 técnicas foi a escolhida.

No *Blur* a técnica *Gaussian* foi escolhida por apresentar resultado superior ao *Motion* e ao *Median* nos 3 critérios, valor mínimo, valor médio e valor máximo.

Nas Transformações Geométricas o foi selecionado *Horizontal*, que se mostrou mais eficaz nos modelos InceptionV3, ResNet e VGG19, perdendo para o *Vertical* nos modelos ResNet50 e VGG16, em relação ao F-Beta médio.

Nas Transformações de Cor as técnicas *Brightness* e *ColorJitter* apresentaram resultados inferiores no F-Beta médio em 4 modelos, são eles o InceptionV3, ResNet50, VGG16 e ResNet, sendo que no geral a queda foi mais acentuada no *Brightness*. No *ColorJitter* houve uma melhoria no VGG19, enquanto que no *Brightness* o VGG19 também perdeu performance. Portanto, a técnica escolhida foi a *ColorJitter*

Os resultados do experimento com a combinação das 4 técnicas estão presentes na Tabela 23

Modelo	**F-Beta(Min)**		**F-Beta(Médio)**		**F-Beta(Max)**	
	Sem DA	**DA**	**Sem DA**	**DA**	**Sem DA**	**DA**
InceptionV3	0.3125	0.357	0.5718	0.5561	0.7291	0.75
ResNet50	0.5208	0.4166	0.7224	0.6571	0.875	0.8333
VGG16	0.6730	0.5555	0.7629	0.7074	0.9090	0.8035
ResNet	0.5357	0.3571	0.6526	0.6202	0.8653	0.8653
VGG19	0.6730	0.7954	0.7955	0.8420	0.875	0.9090

Tabela 23. Resultados dos modelos com as diferentes técnicas combinadas no conjunto de validação.

Analisando a tabela é possível observar que apenas o VGG19 teve um aumento em seu desempenho, de 0.79 para 0.84 em seu F-Beta médio, além de melhorar o F-Beta mínimo e máximo.

4.3. Melhores Resultados

Para o critério de avaliação dos modelos, o F-Beta médio foi a métrica escolhida para definir o desempenho. Os resultados finais podem ser observados na Tabela 24. Ela mostra cada um dos modelos em sua melhor configuração. Uma nova coluna, a *Técnica*, mostra se o melhor resultado foi obtido sem DA, com alguma técnica específica ou com as técnicas combinadas de DA.

Modelo	F-Beta(Min)		F-Beta(Média)		F-Beta(Max)		Técnica
	Sem DA	DA	Sem DA	DA	Sem DA	DA	
InceptionV3	0.3125	0.4166	0.5718	0.6405	0.7291	0.7954	RandomCrop
ResNet50	0.5208	0.6944	0.7224	0.7670	0.875	0.8333	Vertical
VGG16	0.6730	-	0.7629	-	0.9090	-	Sem DA
ResNet	0.5357	-	0.6526	-	0.8653	-	Sem DA
VGG19	0.6730	0.7954	0.7955	0.8420	0.875	0.9090	Téc Comb

Tabela 24. Resultados dos melhores modelos considerando todas as técnicas de DA no conjunto de validação.

Na Tabela 25, os resultados do VGG19 são comparados com o modelo VGG sem alterações (*baseline*) e com o modelo utilizando um *threshold* [5] específico para melhorar os resultados (Mattos e Ribeiro, 2022). Os dois primeiros modelos da tabela utilizam um split de 80%/20%, enquanto que o VGG19 utiliza 60%/20%/20%.

Modelo	Precisão	F-Beta	Recall
VGG - base	0.727	0.701	0.615
VGG - *threshold*	0.909	0.877	0.769
VGG19 - DA	0.951	0.842	0.583

Tabela 25. Resultados do modelo VGG19 com o uso das técnicas de DA combinadas, com o modelo *baseline* e o modelo com *threshold* no conjunto de validação.

Ambos os modelos melhoraram, se comparados com o modelo *baseline*. O VGG19 obteve resultados inferiores ao VGG com *thresold* devido ao *Recall* ser menor, porém a Precisão aumentou, uma métrica importante dado o contexto do problema.

Com os modelos devidamente treinados e sem a previsão de novas modificações no modelo[6], são gerados os resultados do conjunto de dados de teste. Os resultados dão a noção de qual seria o resultado real, caso fosse feita a integração com a plataforma Cancontrol. Os dados de teste não são utilizados durante o treinamento/validação, o que reduz as chances de *overfitting*. O mesmo ocorre com as imagens enviadas à plataforma para diagnóstico pelos produtores, elas não são utilizadas durante essas etapas. No VGG19, os resultados obtidos foram de 1 (100%) para a Precisão, 0.1538 (15.38%) para o *Recall* e 0.4761 (47.61%) para o F-Beta. Esses valores não serão comparados com os dos modelos

[5] O *threshold* é o limite inferior a partir do qual o modelo considera uma imagem como infectada. Nesse caso o limite inferior utilizado é de 0.530

[6] É importante que os dados de teste sejam utilizados apenas quando se há a certeza de que o modelo não será mais modificado. Os dados de teste não podem ser vistos duas vezes.

VGG *baseline* e *threshold*, visto que em ambos os casos a divisão dos dados foi feita com 80% das imagens para treino e 20% para validação. Portanto, não existem dados para teste nestes modelos.

4.4. Considerações Finais

A partir dos resultados obtidos é possível observar que houve uma pequena melhoria em modelos específicos. Uma hipótese que pode justificar a ausência de melhores resultados, como os obtidos nos trabalhos similares apresentados na seção 2.5, é a natureza do problema. É comum utilizar um conjunto de dados onde a doença se encontra na folha ou no fruto da planta, o que gera uma maior variação de cores, por exemplo, facilitando a detecção. No caso do cancro europeu os sintomas estão no galho e no tronco e se manifestam de forma diferente na árvore, dependendo de quanto tempo a planta está infectada com a doença. Nas Figuras 11 e 12 é possível observar exemplos de falso positivo ao lado de um verdadeiro negativo e um verdadeiro positivo com falso negativo, respectivamente. Ao analisar estas imagens nota-se que as diferenças entre árvores saudáveis e árvores infectadas podem ser muito sutis, e portanto, dificilmente podem ser diferenciadas a olho nu.

Figura A Figura B

Figura 11. Galho sem a doença classificado como positivo (A) e galho classificado sem a doença corretamente (B).

Fonte : Imagens retiradas da plataforma Cancontrol.

Figura A Figura B

Figura 12. Galho com a doença classificado corretamente (A) e galho com a doença classificado como negativo (B).

Fonte : Imagens retiradas da plataforma Cancontrol.

5. Conclusão

O cancro europeu é uma doença que vem se espalhando de forma acelerada nos pomares do sul do Brasil. No estado de Santa Catarina os produtores recorrem aos especialistas da Epagri para fazer o diagnóstico de uma amostra da planta. Devido ao alto índice de proliferação da doença, são muitos os casos a serem examinados pelos membros da Epagri, o que vem gerando sobrecarga nos funcionários.

Este trabalho tem como objetivo testar diferentes modelos de CNNs, bem como testar se a implementação de técnicas de *Data Augmentation* podem ser utilizadas para criar modelos robustos, para o diagnóstico automático do Cancro Europeu por meio da análise de imagens, dadas as limitações presentes no conjunto de dados.

Dentre os modelos utilizados, o ResNet e o VGG16, presentes no trabalho de Mattos e Ribeiro (2022) foram mantidos, os demais foram descartados e os modelos InceptionV3, ResNet50 e VGG19 foram adicionados.

Foram realizados vários experimentos com técnicas de DA para aumentar a diversidade dos dados durante o treinamento dos modelos e melhorar sua generalização. Inicialmente foram feitos testes com diversos tipos de DA e também testados diferentes valores nos parâmetros de suas funções a fim de se realizar uma seleção inicial de quais técnicas utilizar e os parâmetros mais adequados para não correr o risco de prejudicar o modelo fazendo com que ele não pudesse classificar as imagens corretamente.

Inicialmente foram realizados testes onde podiam ser aplicadas diversas técnicas de DA em uma mesma imagem, com uma restrição para que somente uma das técnicas de cada grupo dos citados nas Seções 3.4 ocorresse, com intuito de não realizar transformações, como cortes, por exemplo, de forma consecutiva impossibilitando a classificação correta para a imagem em questão. Mesmo com este cuidado, esta abordagem não trouxe bons resultados e por conta disso decidiu-se realizar um novo experimento utilizando somente uma técnica por vez e também apenas a configuração que obteve os melhores resultados para cada modelo. Com esta segunda abordagem os resultados pude-

ram ter um pequeno incremento em alguns dos resultados obtidos anteriormente, os quais foram apresentados nas seções anteriores. Além destes, ainda foi feito um outro experimento utilizando 4 técnicas de DA combinadas, sendo cada uma a que gerou os melhores resultados individualmente dentro dos grupos apresentados anteriormente.

Os resultados apresentados na Seção 4 mostram que o uso das técnicas de DA trouxeram melhorias em alguns modelos, como o VGG19 que atingiu os maiores valores de FBETA, melhorando de 0.7955 para 0.8420 no FBETA médio. Também o ResNet50, que passou de 0.7224 para 0.7670 no FBETA médio e o InceptionV3, que melhorou 0.0687, passando de 0.5718 para 0.6405. Entretanto, devido a baixa quantidade de imagens e os resultados obtidos no teste pode-se perceber que o problema do *overfitting* não pode ser totalmente resolvido, baseado nisso propõe-se sugestões para trabalhos futuros.

Para possíveis trabalhos futuros, testes fatoriais trarão estatísticas mais confiáveis a respeito das técnicas de DA utilizadas, além de trazer dados sobre a interação entre as técnicas. Para melhorar a assertividade, diferentes abordagens podem ser escolhidas.

A utilização de diferentes técnicas e modelos, não apresentados neste trabalho, podem gerar resultados melhores. Também deve ser considerada a hipótese de treinar um modelo do zero, considerando que os modelos utilizados são pré-treinados no conjunto de dados *ImageNet*, composto de 1000 classes diferentes, envolvendo desde animais até a veículos. O conjunto PlantVillage (Hughes e Salathe, 2016) contém apenas imagens de folhas de diferentes espécies de plantas e pode fornecer um conjunto de pesos mais adequado para o problema de detecção do cancro europeu. Quando um modelo for considerado apto, este deve ser integrado a plataforma do Cancontrol para que seja feito o diagnóstico automático.

Referências

Abbas, A., Jain, S., Gour, M., e Vankudothu, S. (2021). Tomato plant disease detection using transfer learning with c-gan synthetic images. *Computers and Electronics in Agriculture*, 187:106279.

Alves, S. A. M., Czermainski, A. B. C., Czermainski, A. B. C., et al. (2019). O cancro europeu no brasil. *Embrapa*.

Amazon (2023). O que é uma rede neural? Acessado em: 17-05-2023.

Arora, S., Bhaskara, A., Ge, R., e Ma, T. (2013). Provable bounds for learning some deep representations.

Branco Neto, W. C., Araújo, L., Pinto, F., Machado, R., Ribeiro, Y., Junior, W. C., e Mattos, K. (2021). Cancontrol: plataforma para diagnóstico do cancro europeu da macieira. In *Anais do XIII Congresso Brasileiro de Agroinformática*, pages 44–52, Porto Alegre, RS, Brasil. SBC.

Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., e Kalinin, A. A. (2020). Albumentations: Fast and flexible image augmentations. *Information*, 11(2).

Goodfellow, I., Bengio, Y., e Courville, A. (2016). *Deep Learning*. MIT Press.

Haba, D. (2023). *Data Augmentation with Python: Enhance deep learning accuracy with data augmentation methods for image, text, audio, and tabular data*. Packt Publishing.

Haykin, S. S. (2009). *Neural networks and learning machines*. Pearson Education, Upper Saddle River, NJ, third edition.

He, K., Zhang, X., Ren, S., e Sun, J. (2015). Deep residual learning for image recognition.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q. V., e Adam, H. (2019). Searching for mobilenetv3.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., e Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications.

Hughes, D. P. e Salathe, M. (2016). An open access repository of images on plant health to enable the development of mobile disease diagnostics.

Khan Ph.D., A., Nawaz, U., Ulhaq, A., e Robinson, R. (2020). Real-time plant health assessment via implementing cloud-based scalable transfer learning on aws deeplens. *PLOS ONE*, 15:e0243243.

Kist, B. B. (2019). Anuário brasileiro da maçã.

Lecun, Y., Bottou, L., Bengio, Y., e Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Liang, Q., Xiang, S., Hu, Y., Coppola, G., Zhang, D., e Sun, W. (2019). Pd2se-net: Computer-assisted plant disease diagnosis and severity estimation network. *Computers and Electronics in Agriculture*, 157:518–529.

Lima, M., Rubik, E., e Morais, R. (2020). ntrodução ao reconhecimento de imagens. `https://lamfo-unb.github.io/2020/12/05/Captcha-Break/`. Acessado em : 23-10-2023.

Mattos, K. M. e Ribeiro, Y. F. B. (2022). Diagnóstico do cancro europeu da macieira com redes neurais convolucionais. IFSC, SC.

mxnet (2023). Image augmentation. `https://mxnet.apache.org/versions/1.5.0/tutorials/gluon/data_augmentation.html`. Acessado em : 06-08-2023.

Negi, C. (2021). Exploring the impact of data augmentation on deep learning models for plant disease detection in plantvillage dataset. *Mathematical Statistician and Engineering Applications*, 70(2):1372–1382.

Nvidia (2023). Chapter 27. motion blur as a post-processing effect. [Online; accessed 9-October-2023].

Russell, S. J. e Norvig, P. (2022). Redes convolucionais. In *Inteligência Artificial - Uma Abordagem Moderna*, pages 687–688. GEN LTC.

Sagar, A. e Jacob, D. (2021). On using transfer learning for plant disease detection. *bioRxiv*.

Shorten, C. e Khoshgoftaar, T. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data, SpringerOpen*.

Simonyan, K. e Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition.

Stanford, U. (2023). Cs231n convolutional neural networks for visual recognition. Acessado em: 16-05-2023.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., e Rabinovich, A. (2014). Going deeper with convolutions.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., e Wojna, Z. (2015). Rethinking the inception architecture for computer vision.

Takahashi, R., Matsubara, T., e Uehara, K. (2020). Data augmentation using random image cropping and patching for deep CNNs. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(9):2917–2931.

TorchVision (2016). TorchVision: PyTorch's Computer Vision library.

Vegas Creative Software (2023). Using motion blur for natural movement. [Online; accessed 9-October-2023].

Wang, C.-F. (2018). A basic introduction to separable convolutions. Acessado em: 15-05-2023.

Weber, R. (2014). Biology and control of the apple canker fungus neonectria ditissima (syn. n. galligena) from a northwestern european perspective. *Erwerbs-Obstbau*, 56:95–107.