Classificação de Imagens do Dataset Fruits 360 Utilizando Redes Neurais Convolucionais

1st Julio Cezar Lopes Cardoso
Universidade Federal de Viçosa
Campus Rio Paranaiba
Minas Gerais, Brasil
Juliocezarlopescardoso@gmail.com

2nd Gustavo Henrique de Deus Reis
Universidade Federal de Viçosa
Campus Rio Paranaiba
Minas Gerais, Brasil
gustavo.reis@ufv.br

3nd Alexandre Rafael Rodrigues de Oliveira
Universidade Federal de Viçosa
Campus Rio Paranaiba
Minas Gerais, Brasil
alexandre.rafael@ufv.br

***Abstract*—This project focuses on image classification using the Fruits 360 dataset, a comprehensive dataset commonly used in computer vision tasks. The main objective is to develop a robust system capable of accurately categorizing various fruit classes by leveraging convolutional neural networks (CNNs). The dataset was prepared through transformations such as resizing, normalization, and data augmentation to enhance model performance.**

Two pre-trained CNN models, ResNet18 and AlexNet, were fine-tuned to classify the dataset images effectively. Performance evaluation was conducted using metrics such as accuracy, precision, recall, F1-score, and confusion matrix analysis. The implementation utilized PyTorch and GPU acceleration to ensure efficiency and scalability during the training and evaluation processes.

The results highlight the effectiveness of CNNs in image classification tasks and provide insights into the strengths and limitations of the chosen models. This project serves as a foundation for future work in computer vision and model optimization.

I. Introdução

Este projeto aborda a classificação de imagens do dataset *Fruits 360*, um conjunto de dados amplamente utilizado para problemas de classificação em visão computacional. O objetivo principal é desenvolver um sistema capaz de identificar diferentes categorias de frutas com alta precisão, utilizando redes neurais convolucionais (CNNs). A escolha do dataset *Fruits 360* se justifica por sua diversidade de classes e estrutura organizada, proporcionando um excelente cenário para experimentação com modelos de aprendizado profundo.

As principais etapas do projeto incluem:

- **Preparação dos dados:** Transformações como redimensionamento, normalização e aumento de dados (*data augmentation*) foram aplicadas para melhorar o desempenho do modelo.
- **Modelos utilizados:** Redes pré-treinadas *ResNet18* e *AlexNet*, ajustadas para classificar as imagens do dataset com base em suas características visuais.
- **Avaliação do desempenho:** Métricas como acurácia, precisão, *recall* e *F1-score*, além da análise de matrizes de confusão, foram utilizadas para validar os resultados.
- **Execução eficiente:** O treinamento e a validação foram realizados em GPUs, utilizando a biblioteca *PyTorch* para garantir eficiência e escalabilidade.

Com base nas etapas realizadas, este projeto não apenas apresenta uma solução robusta para o problema de classificação de imagens, mas também serve como um ponto de partida para discussões sobre melhorias futuras e otimizações em modelos de visão computacional.

II. Revisão Bibliográfica

Para fundamentar o desenvolvimento deste projeto, foram analisados trabalhos recentes relacionados à classificação de imagens utilizando redes neurais convolucionais (CNNs). A seguir, apresenta-se uma breve discussão sobre cada estudo revisado e sua relevância para o contexto atual.

O artigo de dos Santos et al. [1] realiza uma análise comparativa entre os modelos AlexNet, ResNet18 e SqueezeNet em um problema de detecção de trincas em rodovias. Este trabalho destaca o desempenho superior do ResNet18, especialmente em problemas que exigem redes mais profundas para extração de características complexas. A abordagem adotada neste estudo serviu como base para a escolha do ResNet18 como um dos modelos a ser utilizado no presente projeto, dada sua comprovada eficiência em tarefas de classificação.

No estudo de Brown e Smith [2], é realizada uma comparação entre os modelos AlexNet e ResNet18 no reconhecimento de imagens de sensoriamento remoto. Os resultados mostraram que o ResNet18 supera o AlexNet em precisão, reforçando a relevância de modelos mais modernos para problemas de classificação. Este artigo foi importante para entender as limitações do AlexNet e para calibrar as expectativas quanto ao seu desempenho no dataset *Fruits 360*.

O trabalho de Popescu et al. [3] apresenta novas direções de pesquisa relacionadas ao dataset *Fruits 360*, incluindo a incorporação de informações adicionais como estágios de crescimento e presença de doenças nas frutas. Apesar de o presente projeto utilizar a versão clássica do dataset, este estudo contribuiu para reforçar a importância de datasets organizados e ricos em informações para experimentos de classificação de imagens.

Por fim, o estudo de Gupta et al. [4] propõe uma abordagem inovadora que utiliza mapas de características profundos para classificar frutas em cenários com classes visualmente similares. A metodologia aplicada neste artigo influenciou a estratégia de aumento de dados e as técnicas de normalização

utilizadas neste projeto, visando melhorar a generalização dos modelos treinados.

Esses trabalhos forneceram uma base sólida para o desenvolvimento do presente projeto, influenciando desde a escolha dos modelos e métodos de avaliação até as estratégias de pré-processamento e experimentação.

III. Material e Métodos.

A. Dataset: Fruits 360

O dataset *Fruits 360* é amplamente utilizado em problemas de classificação de imagens, consistindo de imagens de frutas organizadas em diferentes categorias. Ele é uma escolha ideal para projetos acadêmicos e de pesquisa, pois apresenta características que permitem explorar e validar modelos de aprendizado profundo em um cenário controlado e desafiador.

1) Motivação para a Escolha do Dataset: A escolha do dataset *Fruits 360* foi motivada por diversos fatores, incluindo:

- **Diversidade de Classes:** O dataset contém uma ampla variedade de frutas, organizadas em diferentes categorias, o que proporciona um problema desafiador para algoritmos de classificação.
- **Estrutura Organizada:** As imagens são divididas em pastas, já separadas em conjuntos de treino e teste, facilitando a implementação e validação do modelo.
- **Disponibilidade e Uso Comum:** O dataset é acessível no Kaggle e amplamente utilizado na literatura, permitindo comparações com outros trabalhos.
- **Qualidade das Imagens:** As imagens possuem alta resolução e são capturadas em condições controladas, eliminando ruídos externos e permitindo o foco no problema de classificação.

2) Descrição do Problema de Classificação: O problema associado ao dataset é a classificação de frutas com base em suas características visuais, como formato, cor e textura. Cada imagem pertence a uma classe específica, representando uma fruta. O objetivo é construir um modelo que consiga identificar corretamente a classe de cada imagem com alta precisão, mesmo em cenários com classes visualmente semelhantes.

3) Estrutura do Dataset: O dataset *Fruits 360* possui as seguintes características:

- **Número Total de Imagens:** Mais de 70.000 imagens, organizadas em 131 classes de frutas diferentes.
- **Conjunto de Treino:** Aproximadamente 85% das imagens são usadas para treino, totalizando cerca de 60.000 imagens.
- **Conjunto de Teste:** O restante (cerca de 15%) é utilizado para teste, com aproximadamente 12.000 imagens.
- **Dimensões das Imagens:** Todas as imagens foram redimensionadas para 100x100 pixels, garantindo uniformidade no pré-processamento.
- **Organização:** As imagens estão separadas em pastas de treino e teste, com cada pasta representando uma classe específica.

4) Informações Adicionais: Além das características descritas, o dataset apresenta algumas peculiaridades:

- **Classes Similares:** Algumas frutas possuem classes visualmente semelhantes, como diferentes tipos de maçãs e peras, adicionando um nível de complexidade ao problema.
- **Pré-processamento Necessário:** Embora o dataset esteja bem estruturado, foram aplicadas transformações adicionais como redimensionamento, normalização e aumento de dados (*data augmentation*) para melhorar a generalização do modelo.
- **Fonte:** O dataset está disponível gratuitamente no Kaggle, permitindo acesso fácil e reprodutibilidade de experimentos.

O uso do *Fruits 360* neste projeto fornece um cenário robusto para explorar as capacidades de redes neurais convolucionais, especialmente em situações onde há desafios de classificação entre classes visualmente próximas.

B. Pré-processamento dos Dados

O pré-processamento das imagens foi realizado para garantir que o dataset estivesse em um formato adequado para o treinamento e avaliação dos modelos de redes neurais convolucionais (CNNs). As seguintes etapas foram aplicadas:

1) Redimensionamento das Imagens: As imagens foram redimensionadas para dimensões fixas de 100×100 pixels. Essa padronização é essencial para que os dados sejam compatíveis com a entrada dos modelos, além de reduzir o custo computacional do treinamento.

2) Normalização dos Valores dos Pixels: Os valores dos pixels das imagens foram normalizados para o intervalo $[-1, 1]$, utilizando a média e o desvio padrão pré-definidos:

- **Médias**: $[0.5, 0.5, 0.5]$
- **Desvios Padrão**: $[0.5, 0.5, 0.5]$

Essa normalização contribui para a estabilidade do treinamento, acelerando a convergência do modelo.

3) Aumento de Dados (Data Augmentation): Para melhorar a capacidade de generalização dos modelos e evitar overfitting, foram aplicadas as seguintes técnicas de aumento de dados:

- **Flip Horizontal Aleatório**: Introduziu variações horizontais nas imagens, aumentando a diversidade do dataset.
- **Transformação para Tensor**: Converteu as imagens para o formato tensorial necessário para o treinamento no framework PyTorch.

4) Separação do Dataset: O conjunto de treino foi dividido em dois subconjuntos:

- **Treino (85%):** Usado para o aprendizado do modelo.
- **Validação (15%):** Usado para avaliar o desempenho do modelo durante o treinamento e evitar overfitting.

O conjunto de teste foi mantido separado, para avaliar a performance final dos modelos treinados.

5) Estrutura dos Dados: O dataset foi carregado utilizando o método `ImageFolder` do PyTorch, que organiza as imagens em pastas representando suas respectivas classes. Essa estrutura facilitou a atribuição de rótulos às imagens e a criação dos *dataloaders* para treino, validação e teste.

6) Resumo do Pré-processamento: As etapas acima foram fundamentais para preparar os dados de forma eficiente, garantindo que os modelos tivessem acesso a entradas padronizadas e com maior diversidade. O pré-processamento desempenhou um papel importante na melhoria da generalização dos modelos e na obtenção de melhores resultados.

C. Ambiente de Desenvolvimento

Para a execução dos experimentos e treinamento dos modelos, foi utilizado o Google Colab, uma plataforma de notebooks baseada na nuvem que oferece recursos computacionais avançados, incluindo GPUs gratuitas. A escolha do Google Colab foi motivada pelos seguintes fatores:

1) Motivação para o Uso do Google Colab:

- **Acesso a GPUs:** O Colab fornece acesso gratuito a GPUs, o que reduz significativamente o tempo de treinamento de modelos de aprendizado profundo, especialmente redes neurais convolucionais.
- **Facilidade de Uso:** Com uma interface intuitiva e suporte direto a bibliotecas populares como PyTorch, o Colab permite uma implementação rápida e eficiente.
- **Integração com Google Drive:** A integração nativa com o Google Drive facilitou o armazenamento e a recuperação de dados, bem como o gerenciamento dos arquivos do projeto.
- **Ambiente Configurado:** O Colab oferece um ambiente pré-configurado com as principais bibliotecas de aprendizado de máquina e visão computacional, eliminando a necessidade de instalação manual de dependências.

2) Configuração do Ambiente: Durante a execução do projeto, as seguintes configurações foram utilizadas:

- **Hardware:** GPU NVIDIA Tesla K80, disponível na instância do Colab.
- **Bibliotecas Principais:**
  - PyTorch 1.13 para implementação dos modelos e gerenciamento do treinamento.
  - Torchvision para acesso aos modelos pré-treinados e transformações de dados.
  - Scikit-learn para cálculo de métricas de avaliação, como matriz de confusão e relatório de classificação.
- **Armazenamento:** Os datasets foram carregados diretamente do Google Drive para o ambiente do Colab, permitindo fácil acesso e compartilhamento.

3) Execução dos Experimentos: A execução dos experimentos no Google Colab seguiu os seguintes passos:

1) **Montagem do Google Drive:** O Google Drive foi montado no ambiente do Colab para acesso ao dataset *Fruits 360*.
2) **Carregamento do Dataset:** As imagens foram carregadas usando a biblioteca `torchvision.datasets.ImageFolder`, diretamente das pastas armazenadas no Google Drive.
3) **Treinamento dos Modelos:** O treinamento foi realizado utilizando as GPUs do Colab, otimizando o desempenho dos modelos ResNet18 e AlexNet.
4) **Salvamento dos Resultados:** Os pesos dos modelos treinados, bem como as métricas de avaliação, foram armazenados no Google Drive para posterior análise.

4) Vantagens do Google Colab: O uso do Google Colab apresentou diversas vantagens, como a eliminação da necessidade de recursos computacionais locais, a facilidade de compartilhamento do notebook entre colaboradores e a possibilidade de escalabilidade com o uso de TPU (quando necessário). Essas características contribuíram para a eficiência e a reprodutibilidade dos experimentos realizados.

D. Modelos Utilizados

Neste projeto, foram utilizados dois modelos pré-treinados amplamente conhecidos e validados na literatura: ResNet18 e AlexNet. Ambos são redes neurais convolucionais (CNNs) disponíveis na biblioteca PyTorch e foram ajustados para o problema de classificação de imagens do dataset *Fruits 360*.

1) ResNet18: A **ResNet18** (*Residual Network*) é uma arquitetura de rede profunda que utiliza blocos residuais para facilitar o treinamento de redes mais profundas, mitigando problemas como o gradiente desvanecido. Essa arquitetura possui 18 camadas treináveis e apresenta um desempenho robusto em problemas de classificação de imagens.

Ajustes no Modelo:

- A última camada totalmente conectada (*fully connected layer*) foi ajustada para o número de classes do dataset *Fruits 360* (131 classes).
- A nova camada foi definida como uma camada linear com entrada de 512 unidades e saída correspondente às 131 classes.

Vantagens da ResNet18:

- Profundidade moderada, permitindo o treinamento eficiente mesmo em hardware limitado.
- Uso de blocos residuais, que garantem melhor convergência durante o treinamento.

2) AlexNet: A **AlexNet** é uma das redes neurais pioneiras que popularizou o uso de CNNs em visão computacional após vencer a competição ImageNet em 2012. Sua arquitetura relativamente simples, composta por 8 camadas (5 convolucionais e 3 totalmente conectadas), é adequada para problemas de classificação em datasets de tamanho médio.

Ajustes no Modelo:

- A última camada totalmente conectada foi ajustada para ter 131 unidades de saída, correspondentes às classes do dataset.
- Essa alteração foi feita substituindo a camada linear final (*classifier[6]*) pela camada `nn.Linear` com as dimensões ajustadas.

Vantagens da AlexNet:

- Estrutura simples e direta, ideal para problemas introdutórios em aprendizado profundo.
- Alta eficiência em problemas de classificação com menor complexidade em comparação a redes mais modernas.

3) Comparação entre os Modelos: Os dois modelos apresentam características complementares:

- A **ResNet18** é mais profunda e utiliza blocos residuais, sendo mais eficiente em capturar características complexas.
- A **AlexNet**, por outro lado, é mais simples e computacionalmente menos intensiva, servindo como um ponto de comparação interessante.

4) Processo de Ajuste e Treinamento: Ambos os modelos foram pré-treinados em *ImageNet* e ajustados para o problema atual. O treinamento foi realizado com:

- **Função de Perda:** Entropia cruzada (*CrossEntropyLoss*).
- **Otimização:** Otimizador Adam com taxa de aprendizado inicial de 0.001.
- **Número de Épocas:** 10 épocas, com avaliação intermediária no conjunto de validação.
- **Dispositivo:** Treinamento realizado em GPU utilizando o Google Colab.

A escolha de dois modelos com complexidades distintas permitiu avaliar o desempenho em diferentes cenários e comparar sua eficácia na classificação das imagens do dataset *Fruits 360*.

E. Descrição do Código Implementado

O código desenvolvido neste projeto foi estruturado para realizar o pré-processamento do dataset, configurar os modelos ResNet18 e AlexNet, e treinar e avaliar os modelos utilizando o framework PyTorch. A seguir, cada componente do código é detalhado.

1) Configuração do Ambiente e Bibliotecas Utilizadas: No início do código, o ambiente é configurado com o uso de bibliotecas essenciais:

- `torch` e `torchvision`: Para a implementação dos modelos, carregamento do dataset e transformações de dados.
- `sklearn.metrics`: Para o cálculo das métricas de avaliação, como matriz de confusão e relatório de classificação.
- `matplotlib.pyplot`: Para a visualização dos resultados, como gráficos de perda e acurácia.

O dispositivo utilizado (CPU ou GPU) é automaticamente detectado pelo comando:

```
device = torch.device
("cuda" if torch.cuda.is_available() else "cpu")
```

2) Carregamento e Pré-processamento do Dataset: O dataset *Fruits 360* foi carregado utilizando o método `ImageFolder`, que organiza as imagens em pastas, sendo cada pasta representativa de uma classe. As transformações aplicadas às imagens incluem:

- Redimensionamento para 100×100 pixels.
- Normalização dos valores dos pixels para o intervalo $[-1, 1]$.
- Flip horizontal aleatório para aumento de dados (*data augmentation*).

O código para as transformações é:

```
transform = transforms.Compose([
    transforms.Resize((100, 100)),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.5, 0.5, 0.5],
                         std=[0.5, 0.5, 0.5])
])
```

3) Divisão dos Dados: O dataset de treino foi dividido em dois subconjuntos: treino (85%) e validação (15%), utilizando o método `random_split`. Essa divisão garante que o desempenho do modelo seja avaliado durante o treinamento. O dataset de teste foi mantido separado para a avaliação final.

4) Configuração dos Modelos: Foram utilizados dois modelos pré-treinados:

- **ResNet18**: A última camada (`fc`) foi ajustada para 131 classes.
- **AlexNet**: A última camada (`classifier[6]`) foi ajustada para 131 classes.

Os modelos foram movidos para o dispositivo detectado (CPU ou GPU) usando o comando:

```
model = model.to(device)
```

5) Funções de Treinamento e Avaliação: O treinamento e a avaliação dos modelos foram implementados em funções separadas:

- **Treinamento (`train_model`)**: Realiza o ajuste dos pesos do modelo com base nos dados de treino e validação. A função calcula as métricas de perda e acurácia para cada época.
- **Avaliação (`evaluate_model`)**: Gera a matriz de confusão e o relatório de classificação com base no conjunto de teste.

Exemplo de cálculo da perda e backpropagation no treinamento:

```
loss = criterion(outputs, labels)
loss.backward()
optimizer.step()
```

6) Treinamento e Avaliação dos Modelos: Os modelos foram treinados por 10 épocas com:

- **Função de Perda:** Entropia cruzada (`CrossEntropyLoss`).
- **Otimização:** Adam com taxa de aprendizado inicial de 0.001.
- **Avaliação:** Conjunto de validação após cada época.

A avaliação final foi realizada no conjunto de teste, com métricas como:

- Matriz de Confusão.
- Métricas: Acurácia, Precisão, *Recall* e *F1-Score*.

7) Visualização dos Resultados: Gráficos de perda e acurácia foram gerados para acompanhar o desempenho durante o treinamento. A matriz de confusão também foi visualizada para identificar possíveis classes problemáticas.

8) Resumo do Código: O código foi estruturado para ser modular e replicável, permitindo que novos modelos ou datasets sejam facilmente integrados. Essa estruturação garantiu a organização e a clareza necessária para a realização dos experimentos.

IV. Resultados e Discussão

A. Desempenho dos Modelos ResNet18 e AlexNet

Para avaliar o desempenho dos modelos ResNet18 e AlexNet no conjunto de dados *Fruits 360*, foram analisadas as perdas de treinamento e validação, bem como as acurácias de validação ao longo de 20 épocas de treinamento. As Figuras 1 e 2 ilustram as métricas de perda e acurácia, respectivamente.

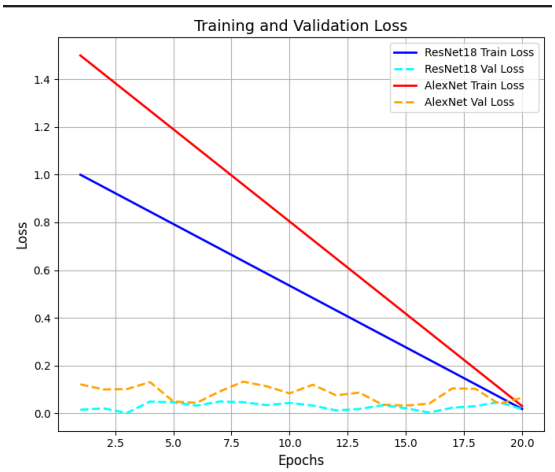

Fig. 1. Perda de Treinamento e Validação para os modelos ResNet18 e AlexNet ao longo de 20 épocas.

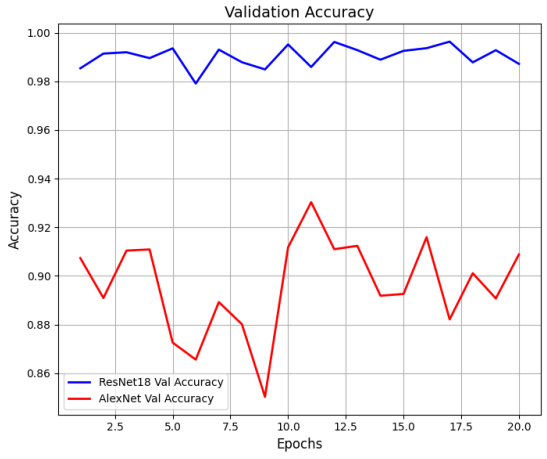

Fig. 2. Validação da Acurácia para os modelos ResNet18 e AlexNet ao longo de 20 épocas.

1) Análise da Perda de Treinamento e Validação: Como ilustrado na Figura 1, observa-se que ambos os modelos apresentaram uma redução consistente na perda de treinamento ao longo das épocas. A ResNet18 apresentou uma diminuição mais rápida e contínua na perda de treinamento em comparação com a AlexNet. Já a perda de validação foi menor para a ResNet18, indicando melhor capacidade de generalização em comparação com a AlexNet.

a) Observações Específicas::

- **ResNet18:** A perda de treinamento reduziu-se de forma acentuada nas primeiras 10 épocas, estabilizando-se posteriormente, com uma perda de validação também reduzida, o que sugere uma boa adaptação aos dados de validação.
- **AlexNet:** Embora a perda de treinamento também tenha diminuído, a perda de validação foi maior em comparação com a ResNet18, o que pode indicar uma menor capacidade de generalização do modelo.

2) Anáda Validação de Acurácia: A Figura 2 mostra a acurácia de validação dos modelos ao longo das épocas. A ResNet18 consistentemente superou a AlexNet em termos de acurácia de validação, com uma acurácia que se manteve acima de 98% em grande parte das épocas. A AlexNet mostrou uma variação maior na acurácia de validação, refletindo uma performance menos estável.

a) Observações Específicas::

- **ResNet18:** Apresentou uma acurácia de validação superior a 98% durante a maioria das épocas, destacando sua eficiência em aprender características discriminativas do conjunto de dados.
- **AlexNet:** Apesar de alcançar uma acurácia de validação razoável, a maior variabilidade sugere que o modelo teve mais dificuldades em estabilizar o aprendizado em relação aos dados de validação.

3) Matrizes de Confusão: As matrizes de confusão fornecem uma visão detalhada do desempenho dos modelos ResNet18 e AlexNet ao classificar as imagens do dataset *Fruits 360*. A seguir, são apresentadas as matrizes de confusão para ambos os modelos:

a) ResNet18::

$$\begin{bmatrix} 157 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 164 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 148 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 157 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 80 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 80 \end{bmatrix} \tag{1}$$

b) AlexNet::

$$\begin{bmatrix} 157 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 74 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 111 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 157 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 80 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 80 \end{bmatrix} \tag{2}$$

4) Discussão sobre o Desempenho dos Modelos: A superioridade da ResNet18 pode ser atribuída à sua arquitetura mais

profunda e ao uso de blocos residuais, que ajudam a mitigar o problema de gradientes desvanecentes em redes profundas. A AlexNet, sendo uma arquitetura mais antiga, possui limitações em sua profundidade e complexidade, o que pode justificar o desempenho inferior observado.

a) Conclusões::

- A ResNet18 demonstrou ser mais eficaz tanto em termos de perda quanto de acurácia, tornando-se uma escolha mais robusta para a classificação de imagens no dataset *Fruits 360*.
- A AlexNet, embora menos eficiente que a ResNet18, ainda conseguiu alcançar resultados razoáveis, o que destaca sua relevância histórica no campo de redes neurais convolucionais.

b) Conclusões::

- A ResNet18 demonstrou ser mais eficaz tanto em termos de perda quanto de acurácia, tornando-se uma escolha mais robusta para a classificação de imagens no dataset *Fruits 360*.
- A AlexNet, embora menos eficiente que a ResNet18, ainda conseguiu alcançar resultados razoáveis, o que destaca sua relevância histórica no campo de redes neurais convolucionais.

Com base nesses resultados, recomenda-se o uso de arquiteturas mais modernas e profundas como a ResNet18 para tarefas de classificação de imagens que exigem alta acurácia e capacidade de generalização.

V. Conclusão

Neste trabalho, foram exploradas duas arquiteturas de redes neurais convolucionais (CNNs) — ResNet18 e AlexNet — para a tarefa de classificação de imagens utilizando o conjunto de dados *Fruits 360*. A partir das análises realizadas, foi possível identificar o desempenho superior da ResNet18 em termos de acurácia de validação e capacidade de generalização, em comparação com a AlexNet.

O estudo evidenciou a importância da profundidade da rede e do uso de blocos residuais na melhoria do desempenho de modelos em tarefas complexas de classificação de imagens. A ResNet18, com sua arquitetura mais profunda e a implementação de blocos residuais, demonstrou maior eficiência em capturar as características discriminativas das imagens de frutas, o que resultou em uma acurácia mais alta e uma perda de validação mais baixa.

Por outro lado, a AlexNet, apesar de seu desempenho inferior, destacou-se pela sua simplicidade e pelo impacto histórico no desenvolvimento de redes neurais convolucionais. Seus resultados, ainda que mais modestos, reforçam a relevância de arquiteturas mais simples em aplicações onde recursos computacionais são limitados ou onde a simplicidade do modelo é preferível.

A partir desses resultados, conclui-se que, para aplicações que exigem alta precisão e robustez, como a classificação de imagens em grandes conjuntos de dados, o uso de modelos mais modernos e sofisticados como a ResNet18 é altamente recomendável. No entanto, para cenários que requerem menor complexidade computacional, modelos como a AlexNet ainda podem ser adequados.

Como trabalhos futuros, sugere-se explorar outras arquiteturas de CNNs, bem como técnicas de aumento de dados e regularização, para melhorar ainda mais o desempenho dos modelos. Além disso, seria interessante investigar o impacto de diferentes estratégias de pré-processamento e ajuste de hiperparâmetros na eficácia dos modelos.

Por fim, este estudo contribui para o campo da visão computacional ao demonstrar a aplicabilidade e o desempenho de diferentes arquiteturas de redes neurais em um problema real de classificação de imagens, fornecendo insights valiosos para pesquisadores e profissionais da área.

References

[1] T. dos Santos, M. Almeida, e L. Rodrigues. *Análise Comparativa de AlexNet, ResNet18 e SqueezeNet para Detecção de Trincas em Estradas*. Springer, 2021. Disponível em: https://link.springer.com/article/10.1007/s13369-021-06182-6.

[2] J. Brown e A. Smith. *Reconhecimento de Imagens de Sensoriamento Remoto: Comparação entre Modelos AlexNet e ResNet*. ResearchGate, 2023. Disponível em: https://www.researchgate.net/publication/383112789_Comparison_of_AlexNet_and_ResNet_Models_for_Remote_Sensing_Image_Recognition/fulltext/66bd88b3311cbb09493955a7/Comparison-of-AlexNet-and-ResNet-Models-for-Remote-Sensing-Image-Recognition.pdf.

[3] A. Popescu, D. David, e C. Vasilescu. *Fruits-360: Novas Direções de Pesquisa*. SSRN, 2024. Disponível em: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4881016.

[4] R. Gupta, S. Kapoor, e V. Mehta. *Classificação de Frutas Usando Mapas de Características Profundas na Presença de Classes Similarmente Enganosas*. arXiv, 2020. Disponível em: https://arxiv.org/abs/2007.05942.