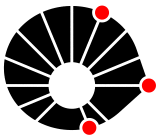

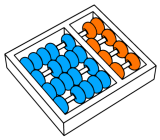

Classificação de Espécies de Pássaros Utilizando Aprendizado Profundo

J.K.R. Matsoui, H. Pedrini

Relatório Técnico - IC-PFG-22-16
Projeto Final de Graduação
2022 - Julho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Classificação de Espécies de Pássaros Utilizando Aprendizado Profundo

Julio Kiyoshi Rodrigues Matsoui* Helio Pedrini†

Julho de 2022

Resumo

Este projeto final de graduação visa estudar e aplicar técnicas da área de processamento de imagens e visão computacional para abordar o problema de classificação de espécies de pássaros. Para isto, a base de dados selecionada para o estudo de casos foi a *BIRDS 400 - Species Image Classification*, disponível no repositório do Kaggle e empregada para treinar os modelos construídos durante o projeto. Para a realização do trabalho, técnicas de processamento e análise de imagens foram investigadas e aplicadas ao problema de classificação. Além disso, técnicas de transferência de aprendizado e aumentação de dados foram exploradas. Resultados experimentais foram coletados para demonstrar a eficácia dos modelos construídos.

1 Introdução

O problema de classificação de espécies de pássaros [9, 11–13, 20, 21, 24] tem sido empregado em diversas áreas do conhecimento, por exemplo, na biologia, em que o estudo sobre a evolução das espécies nativas está sendo auxiliada por redes neurais para detectar as espécies e assim estudar a evolução destas.

Este projeto está inserido no contexto das áreas de processamento de imagens, visão computacional e aprendizado de máquina para auxiliar o desenvolvimento de uma metodologia experimental para a classificação de espécies de pássaros por meio de redes neurais.

Para desenvolver o projeto, primeiramente foi necessário um estudo em técnicas de aprendizado de máquina. Posteriormente, vários artigos da literatura foram revisados para compreender aspectos relevantes utilizados na classificação de imagens, tais como transferência de aprendizado, aumentação de dados, uso de redes pré-treinadas. Vários experimentos foram realizados para cumprir os objetivos previstos.

O trabalho tem o principal intuito de investigar o uso de redes neurais pré-treinadas com ou sem a aplicação de aumentação de dados sobre o conjunto de imagens disponibilizadas no Kaggle [1].

*Instituto de Computação, Universidade Estadual de Campinas, 13083-852, Campinas, SP.
†Instituto de Computação, Universidade Estadual de Campinas, 13083-852, Campinas, SP.

As seções seguintes deste relatório estão organizadas como segue. A Seção 2 efetua uma síntese da revisão da literatura, apresentando estudos relacionados ao tópico investigado. A Seção 3 descreve alguns conceitos utilizados durante os experimentos. A Seção 4 aborda o problema central do projeto e discute a solução para o problema. A Seção 5 descreve os resultados obtidos a partir dos experimentos realizados. Finalmente, a Seção 6 apresenta a conclusão do trabalho.

2 Revisão da Literatura

Esta seção brevemente descreve abordagens existentes na literatura para os problemas de classificação de espécies de pássaros e técnicas de aprendizado profundo. Primeiramente, uma revisão nos conceitos sobre redes neurais convolucionais é realizada, os quais serão empregados para desenvolver o projeto. Posteriormente, uma revisão sobre o problema de classificação de espécies de pássaros é apresentada, bem como as técnicas que foram empregadas para realização deste trabalho.

Um estudo sobre redes convolucionais feito por Zeiler and Fergus [25] apresentou uma forma de visualizar o aprendizado por cada camada da rede neural durante a fase de treinamento. Isto foi importante para o estudo em redes convolucionais, pois mostrou-se que, a partir desta visualização, é possível detectar algum problema com o modelo na etapa de treinamento, ou seja, verificar o que cada camada está "aprendendo" pelo método. Dessa forma, pode-se melhorar o modelo adicionando ou removendo camadas, buscando-se obter melhores resultado.

Para criar este modelo de visualização, é necessário criar uma rede de deconvolução (*Deconvnet*), ou seja, ele aplicará as funções inversas de uma rede convolucional, ou seja, *Unpooling*, camada de retificação (*Rectification*) e, por fim, a filtragem convolucional (*Convolutional Filtering*).

No estado da arte sobre visão computacional, surgiu um aumento no uso de redes convolucionais para tratar problemas de reconhecimento de imagens. Para isto, Chatfield et al. [6] fizeram um estudo detalhado em redes neurais convolucionais, comparando diferentes recursos e esquemas de aumento de dados para identificar estratégias para melhorar o desempenho dessas redes neurais. Para isto, investigaram, como base de seus experimentos, as seguintes redes neurais: *Improved Fisher Vector* (IFV) e três tipos de rede neural convolucional (*convolutional neural network* - CNN), sendo elas uma rápida, uma intermediária e uma lenta em relação ao tempo de treinamento. A conclusão foi que, com o uso de aumento de dados, houve uma melhora significativa no desempenho das redes neurais testadas em um conjunto com poucas imagens.

Nos estudos relacionados à dificuldade em criar uma rede neural com um vasto conjunto de imagens, há o trabalho de Krizhevsky et al. [14], em que se criou uma rede convolucional sobre a base ImageNet [17]. O trabalho abordou a estrutura da rede neural, a estrutura da base ImageNet, as otimizações utilizando placas gráficas para reduzir o tempo de treinamento e a utilização de aumento de dados para reduzir o sobre-ajuste (*overfitting*) nos dados. Ele é um trabalho importante, pois mostra que é possível criar uma rede neural sobre um grande conjunto de imagens, o qual contém 15 milhões de imagens de alta resolução

rotuladas em 22.000 classes.

Uma ferramenta desenvolvida pela Google Brains é o TensorFlow[1], em que Abadi et al. [3] fazem uma introdução sobre ela, mencionando o motivo pelo qual o TensorFlow é uma ferramenta apropriada para o uso em aprendizado de máquina. Algumas características que se julgou importante para um sistema de aprendizado de máquina são a execução distribuída, suporte para aceleradores como unidades gráficas de processamento (*graphics processing unit* - GPU) e unidades de processamento de tensor (*tensor processing unit* - TPU), flexibilidade para se criar o modelo da rede neural, realização de treinamentos e de inferências sobre o modelo e, por fim, o TensorFlow é um projeto de código aberto. Entretanto, há outras ferramentas de aprendizado de máquina além do Tensorflow, tais como Theano [4] e Caffe [18]. Diferentemente dessas ferramentas apresentadas, há o PyTorch, que também permite a criação de arquiteturas de rede neural. O trabalho de Paszke et al. [16] descreve a execução de cálculos de tensores dinâmicos com diferenciação automática e também tem implementações para uso em GPU.

Com relação ao problema de classificação de pássaros, o trabalho de Ferreira et al. [9] fez um estudo populacional de pássaros utilizando-se de uma rede neural para a identificação das espécies de pássaros na reserva florestal de Benfontein, África do Sul. Para coletar as imagens, foi utilizado um aparato de alimentação de pássaros ligado a um sistema de câmeras que, por sua vez, estão conectadas a um sensor criado em uma Raspberry Pi[2] com uma micro-controladora, ou seja, tornando possível acionar a máquina fotográfica e, por sua vez, enviar a fotografia para um banco de dados. Estas fotografias arquivadas no banco de dados são utilizadas para treinar ou re-treinar o modelo. O modelo é utilizado para fazer inferências sob as espécies locais para realizar o controle populacional da reserva. Para criar o modelo, utilizou-se uma rede pré-treinada, VGG-19 [19] por meio da técnica de transferência de aprendizado e, na etapa de pré-processamento das imagens, utilizou-se a técnica de aumento de dados por meio de rotações aleatórias variando de 0° a 40° e ampliações de imagens em 20%, para todas as espécies de pássaros.

O trabalho de Huang and Basanta [11] criou um aplicativo para celular, chamado de *Internet of Birds* (IoB)[3], para a identificação de pássaros endêmicos na região de Taiwan. O conjunto de imagens utilizado para treinar a rede neural contém 27 classes de pássaros. A etapa de pré-processamento de dados utilizou a técnica de aumento de dados, empregando inversões horizontais e verticais (*horizontally and vertically flipped*), rotações de no máximo 25°. Outra técnica usada foi a translação da imagens, translação em -10 até 10 pixels e, por último, a filtragem gaussiana. Para a criação da rede neural, usou-se a API do Keras [8] criando e treinando sua CNN e assim disponibilizando o modelo para fazer inferências. Por fim, para criar o sistema, desenvolveu-se um sistema cliente-servidor para possibilitar aos usuários fazerem *upload* das fotografias, de forma que o servidor possa obter as imagens e fazer a inferência com o modelo treinado.

Outro trabalho de classificação de pássaros foi reportado por Kamlesh Borana and Rajdeep Sodha [13], cuja estratégia foi reconhecer o pássaro por meio de uma rede Mask R-CNN [10] pré-treinada. Também utiliza a estratégia de aumento de dados por meio da

[1] `https://www.tensorflow.org/`
[2] `https://www.raspberrypi.org/`
[3] `http://internetofbirds.com/`

escala de cinza, redimensionamento aleatório das imagens, cortes na imagem e rotações. A utilização da rede Mask R-CNN aumentou a precisão do treinamento, já que a rede é usada para detectar objetos, no caso, pássaros.

Recentemente, alguns projetos de classificação de pássaros baseiam-se em cantos de pássaros. Um trabalho desenvolvido por [24] utilizou estes cantos de pássaros para realizar a classificação de espécies. Para isto, criou-se uma rede CNN de classificação acústica, no qual as entradas são ondas unidimensionais, no qual posteriormente o sinal passa por outras camadas de redes neurais considerando informações de tempo e frequência a partir de uma transformada de Fourier.

Outros trabalhos que utilizam cantos de pássaros para classificação de espécies foram propostos por Kahl et al. [12], Stowell et al. [22], Song and Li [20] e Stowell and Plumbley [21]. Entretanto, um problema recorrente nos métodos de classificação é que, para obter uma base de dados adequada, são necessários longos períodos de gravações para obter cantos de pássaros com boa qualidade. O processo de gravação dos cantos é tipicamente sujeito a muitos ruídos.

2.1 Arquiteturas CNNs

Quatro arquiteturas de redes neurais convolucionais foram selecionadas da literatura para construir os modelos de classificação durante os experimentos realizados neste trabalho, mostrados na Seção 4.

As próximas subseções descrevem as arquiteturas VGG-16, VGG-19, Inception V3 e Xception.

2.1.1 VGG

A rede neural VGG (*Virtual Geometric Group*) foi proposta no trabalho de Simonyan and Zisserman [19] e criada para competir no desafio ILSVRC (*ImageNet Large Scale Visual Recognition Challenge*) de 2014. A arquitetura da VGG é baseada em camadas convolucionais e camadas totalmente conectadas, utilizando-se de maxPooling e Softmax. A arquitetura VGG-16 tem 13 camadas convolucionais e 3 camadas totalmente conectadas, enquanto a VGG-19 tem 16 camadas convolucionais e 3 camadas totalmente conectadas.

A arquitetura VGG-16 obteve resultados melhores no desafio ILSVRC do que a arquitetura VGG-19. As arquiteturas VGG-16 e VGG-19 são mostradas na Figura 1, em que *conv* é a camada convolucional e *FC* é a camada totalmente conectada.

2.1.2 Inception

A arquitetura Inception foi introduzida por Szegedy et al. [23], em que, ao invés de utilizar várias camadas sequenciais para tornar o modelo profundo, empregou-se a estratégia de adicionar várias camadas em um mesmo nível, ou seja, utiliza-se de camadas em paralelo. Dessa maneira, prefere-se uma arquitetura mais ampla, ao invés de uma arquitetura mais profunda.

A arquitetura da Inception-V1 é baseada nos módulos inception, mostrados na Figura 2. Para as outras versões, Inception-V2 e Inception-V3, tem-se uma melhoria nos módulos

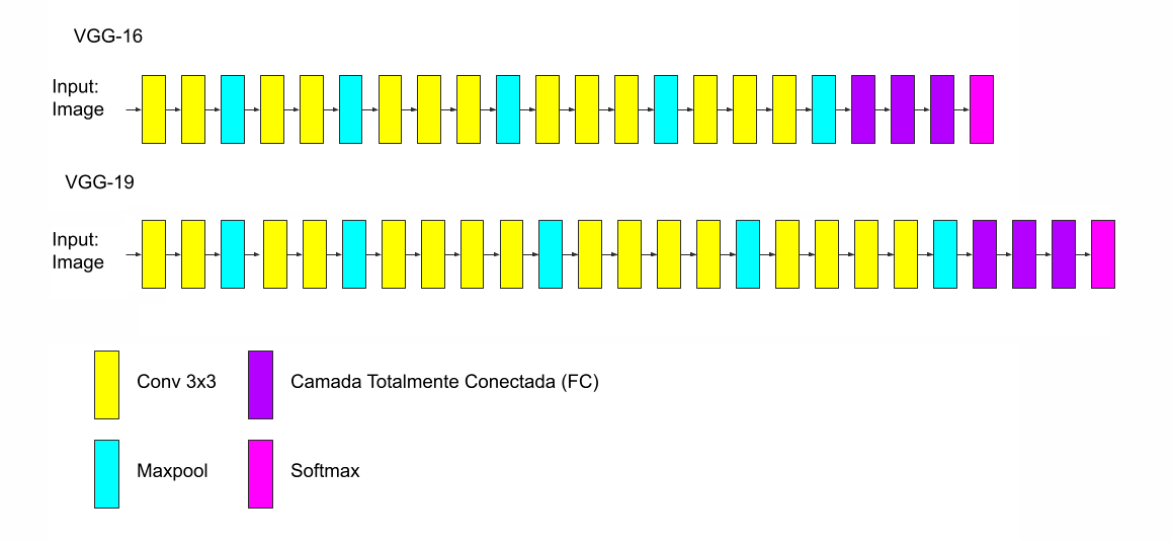

Figura 1: Arquitetura da rede VGG. Figura adaptada de [19].

inceptions para tornar as redes mais largas e mais profundas. A arquitetura da Inception-V3 é constituída de camadas convolucionas, módulos inceptions, maxPooling e Softmax, sendo estas formadas de *hidden layers 22*.

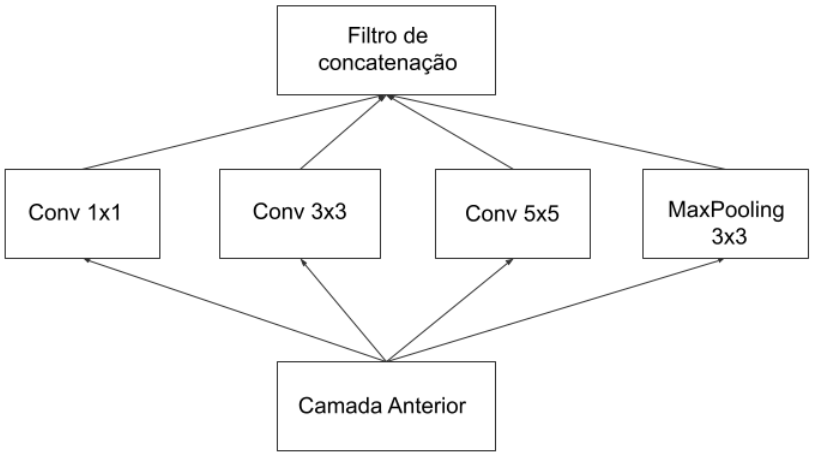

Figura 2: Módulo inception. Figura adaptada de [23].

2.1.3 Xception

A arquitetura Xception foi introduzida por Chollet [7], a qual utiliza a mesma estratégia da arquitetura Inception, no que diz respeito ao uso de módulos. Nela, há os módulos Xceptions, mostrados na Figura 3. A arquiterura Xception é composta por 36 camadas de

convolução estruturadas em 14 módulos Xception.

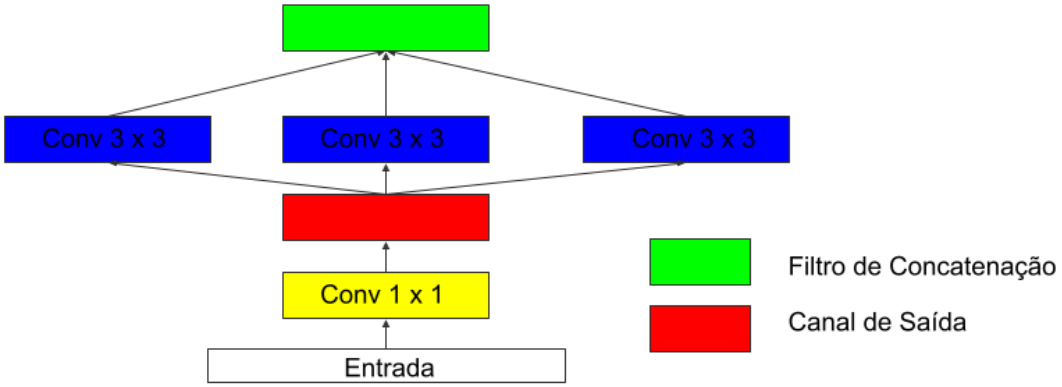

Figura 3: Xception Module. Figura adaptada de [7].

3 Conceitos Relacionados

Nesta seção, alguns conceitos relevantes relacionados ao desenvolvimento deste trabalho são descritos. Informações detalhadas a respeito desses conceitos podem ser encontrados no trabalho de Almeida [5].

- Base de dados: neste trabalho, refere-se a um conjunto de imagens dividido em três partes: conjunto de dados para treinamento da rede neural, conjunto de dados para validar o treinamento e o conjunto de dados para testar o modelo.
- Camadas da rede neural: as camadas de uma rede neural são formadas por um conjunto de neurônios. A primeira camada é chamada de entrada (*input*, a última camada é chamada de *output* e as camadas intermediárias são chamadas de ocultas (*hidden layers*), sendo estas responsáveis por extrair as características das imagens.
- Rede neural convolucional: refere-se a um algoritmo de aprendizado profundo (*deep learning*) que pode capturar várias características em uma imagem de entrada, tipicamente com o intuito de diferenciar os objetos presentes nas imagens.
- Camada de *pooling*: utilizada para reduzir as dimensões dos dados. A forma de agregação de pixels mais comum é conhecida como *MaxPooling*, que seleciona o pixel com maior intensidade da região. A operação reduz a variância da rede a pequenas alterações e diminui a quantidade de parâmetros, agilizando o treinamento da rede neural.
- Camada de *dropout*: realiza uma regularização na rede, tal que alguns neurônios durante o treinamento são desligados aleatoriamente, juntamente com suas conexões, para reduzir o efeito de *overfitting* no treinamento. Durante a predição, os neurônios são mantidos ativos.

- Camada *flatten*: a matriz resultante das camadas anteriores de convolução e de *pooling* é redimensionada nesta camada para se tornar um vetor (*array* com uma única dimensão. Esta etapa é normalmente utilizada na transição das camadas de convolução para a camada totalmente conectada da rede neural.
- Camada densa: implementada por uma operação linear para gerar a saída por uma função baseada em cada entrada. No contexto de reconhecimento de imagens, é comum se projetar esta camada densa com a função de ativação ReLU e, por fim, outra camada densa com a dimensão igual à quantidade de classes a serem classificadas com a função de ativação Softmax (para múltiplas classes).
- Função de ativação Softmax: função que transforma as saídas para cada classe em valores de probabilidade (intervalo entre 0 e 1) para lidar com problemas multiclasse, em contraste com a função sigmoide, que lida com problemas binários.
- Transferência de aprendizado: técnica em que um modelo que já foi treinado para resolver um problema específico é reutilizado para a solução de um novo problema relacionando ao que já foi resolvido [2, 15]. Neste trabalho, como a base utilizada é composta por poucos dados, algumas camadas foram utilizadas, geralmente de identificação de objetos destas redes que já foram treinadas.
- Ajuste fino em redes pré-treinadas: uso de camadas mais rasas de uma rede pré-treinada, eliminando as camadas de classificação e de saída. Pode-se escolher quais camadas da rede pré-treinada devem ser congeladas (neste contexto, significa que os pesos das camadas congeladas não alterarão de acordo com o treinamento). Assim, o ajuste fino (*fine tuning*) significa considerar pesos de uma rede pré-treinada e usá-la como inicialização para um novo modelo sendo treinado em dados de um domínio similar. Esse procedimento é normalmente empregado para acelerar o treinamento e superar o tamanho pequeno de um conjunto de dados.
- Sobre-ajuste: fenômeno que ocorre quando um modelo aprende detalhes e o ruído presentes nos dados de treinamento, entretanto, que afeta negativamente o desempenho do modelo em novos dados. Uma forma para contornar o problema de sobre-ajuste (*overfitting*) é a regularização.
- Acurácia: medida que representa a percentagem de elementos classificados corretamente pelo modelo, indicando uma eficácia geral do classificador.
- Aumentação de dados: técnica empregada para aumentar a quantidade de dados e evitar o fenômeno de sobre-ajuste. Exemplos de operações utilizadas para aumentação de dados são transformações de escala, rotação, espelhamento, recortes, inserção de ruídos, alterações de contraste.

4 Metodologia

Como descrito nas seções anteriores, este trabalho iniciou com um estudo geral sobre técnicas de aprendizado de máquina e análise de imagens. Nesta seção, o problema de classificação

de espécies de pássaros é apresentado, a base de dados escolhida para o estudo de caso é descrita e as técnicas aplicadas para abordar o problema são apresentadas.

4.1 Base de Dados

A base de dados escolhida foi a *BIRDS 400 - Species Image Classification* disponível no repositório do Kaggle[4]. O conjunto de dados contém 400 espécies de pássaros e um total de 62.388 imagens de pássaros.

As imagens são subdivididas em 58.388 imagens para a etapa de treinamento do modelo, 2.000 imagens para a validação do modelo e 2.000 imagens para testar o modelo. Vale ressaltar que o número de imagens por classe de pássaros no conjunto de treinamento é desbalanceado, podendo variar entre 133 imagens até 188 imagens por classe.

Figura 4: Exemplo de imagens de espécies de pássaros contidas na base de dados. Adaptado de Kaggle [1].

A Figura 4 mostra algumas amostras de espécies de pássaros contidas na base de dados. A legenda indica o rótulo da classe e a dimensão da imagem.

4.2 Pré-processamento

Nesta etapa de pré-processamento, técnicas de aumentação de dados (Seção 3) foram aplicadas nas imagens da base de dados para melhorar o desempenho da rede na etapa de treinamento.

As técnicas de aumentação de dados escolhidas foram:

- Redimensionamento das imagens: deixa os valores dos pixels entre [0-1], utilizando a biblioteca do Keras (*tf.keras.layers.Rescaling()*). O resultado é mostrado na Figura 5.
- Rotação das imagens: aplica transformações de rotação aleatórias, entre [0°- 360°], utilizando a biblioteca do Keras (*tf.keras.layers.RandomRotation()*). O resultado é mostrado na Figura 6.
- Espelhamento das imagens: aplica aleatoriamente reflexões horizontais e verticais nas imagens, utilizando a biblioteca do Keras (*tf.keras.layers.RandomFlip()*). O resultado do espelhamento aleatório horizontal é mostrado na Figura 7 e o resultado do espelhamento aleatório vertical é mostrado na Figura 8.

[4] `https://www.kaggle.com/datasets/gpiosenka/100-bird-species`

Figura 5: Imagens originais (primeira linha) e resultados após o redimensionamento das imagens (segunda linha). Adaptado de Kaggle [1].

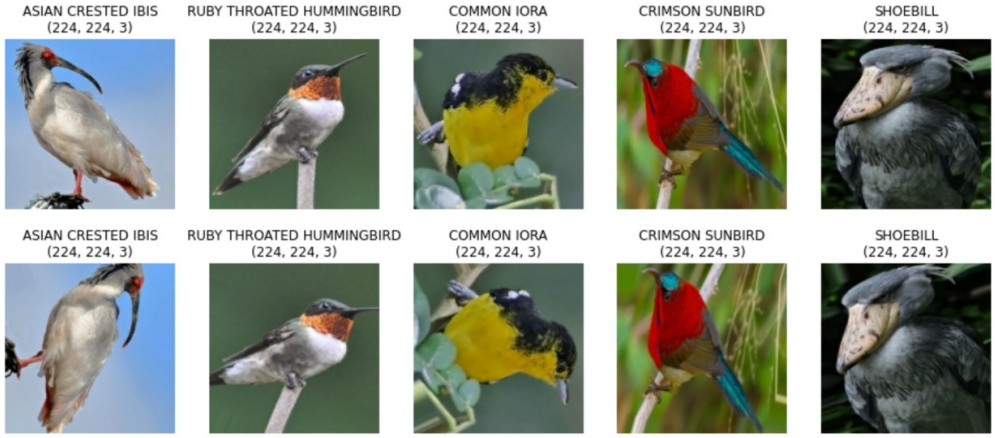

Figura 6: Imagens originais (primeira linha) e resultados após a rotação aleatória das imagens. Adaptado de Kaggle [1].

- Ampliação das imagens: aplica a técnica de ampliação das imagens, utilizando a biblioteca do Keras (*tf.keras.layers.RandomZoom()*). O resultado é mostrado na Figura 9.

4.3 Experimentos

Nesta seção, os experimentos são definidos para a obtenção de resultados. Para a realização dos experimentos, técnicas de aumentação de dados e transferência de aprendizado.

Para realizar os experimentos. dois servidores foram empregados, o Kaggle e o Google Colab, tornando possível utilizar placas gráficas em ambos ambientes. O uso dos servidores para executar os códigos reduziu significativamente o tempo de treinamento das redes. Algumas informações sobre os servidores: placa de vídeo Tesla P100 de 16 GB e processador Intel(R) Xeon(R) CPU @ 2.00GHz.

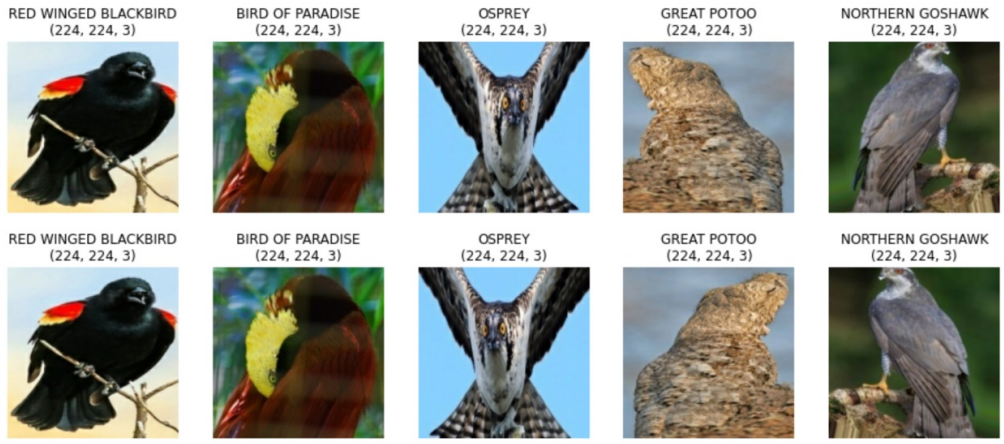

Figura 7: Imagens originais (primeira linha) e resultados após o espelhamento horizontal das imagens. Adaptado de Kaggle [1].

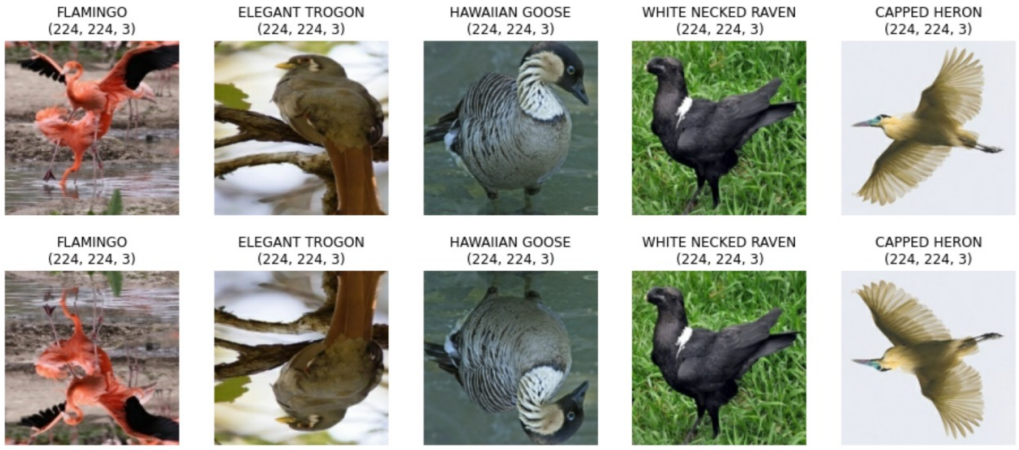

Figura 8: Imagens originais (primeira linha) e resultados após o espelhamento vertical das imagens. Adaptado de Kaggle [1].

Para a realização dos experimentos, uma visão geral da arquitetura da rede é ilustrada na Figura 10. A camada na cor cinza é opcional, ou seja, ao longo dos experimentos poderá ser usada ou não. A camada na cor violeta representa a rede neural pré-treinada, que pode ser VGG-16, VGG-19, Inception V3 e Xception. As outras camadas já foram explicadas na Seção 3.

Para compilar o modelo, o otimizador *Adam* foi empregado, com função de erro *CategoricalCrossentropy* e métrica de acurácia. Para treinar o modelo, 20 épocas foram executadas com o parâmetro de *callback* *ReduceLROnPlateau*, que tem a função de monitorar a curva de aprendizado e, caso a curva comece a decrescer, o treinamento é encerrado. Nos casos em que a acurácia estiver abaixo de 65%, a técnica de ajuste fino será utilizada com o intuito

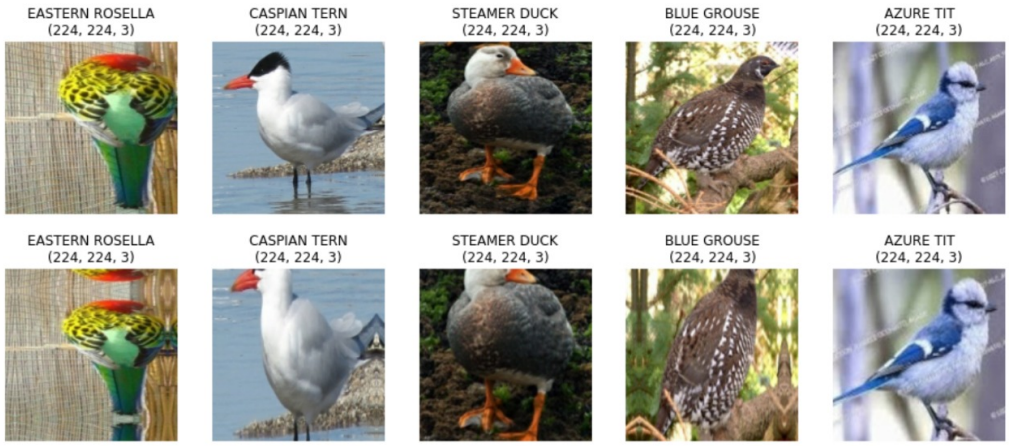

Figura 9: Imagens originais (primeira linha) e resultados após a ampliação das imagens. Adaptado de Kaggle [1].

de melhorar a acurácia do modelo. Os resultados são mostrados nas tabelas da próxima seção, cujos valores foram obtidos na última época treinada.

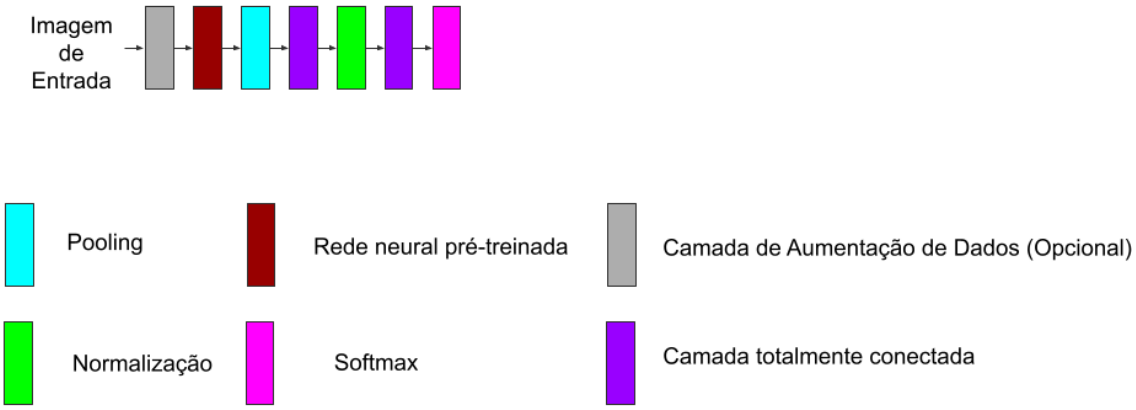

Figura 10: Arquitetura utilizada na realização dos experimentos.

5 Resultados

Esta seção apresenta os resultados de quatro experimentos realizados neste trabalho, cujo objetivo é mostrar qual a melhor técnica para a aumentação de dados e a transferência de aprendizado. As tabelas a seguir apresentam os resultados com base na métrica de acurácia obtidos no conjunto de dados, cujos valores são dispostos como segue. Nos experimentos 1 e 2, a primeira coluna representa a rede neural utilizada, a segunda coluna representa a acurácia no conjunto de treinamento, a terceira coluna representa a acurácia no conjunto de

validação e a quarta coluna representa a acurácia no conjunto de teste. No experimento 3, foi necessário utilizar a técnica de ajuste fino, sendo que, na coluna correspondente, tem-se o valor de acurácia de treinamento (acc) e a acurácia de validação ((val_acc). No experimento 4, a primeira coluna representa a técnica de aumentação de dados utilizada, enquanto as demais colunas apresentam os valores de acurácia para os conjuntos de treinamento, validação e teste, respectivamente.

5.1 Experimento 1

No experimento 1, as técnicas de aumentação de dados não foram aplicadas durante a etapa de treinamento. Os resultados obtidos neste experimento para diferentes arquiteturas são apresentados na Tabela 1.

Tabela 1: Resultados obtidos no experimento 1.

Redes Neurais	Acurácia de Treinamento	Acurácia de Validação	Acurácia de Teste
VGG19	0,9852	0,8975	0,9620
VGG16	0,9859	0,8995	0,9791
Inception-V3	0,8241	0,8790	0,8120
Xception	0,9388	0,9488	0,9071

5.2 Experimento 2

No experimento 2, as seguintes técnicas de aumentação de dados foram aplicadas às imagens da base: rotação e espelhamento horizontal. Os resultados obtidos neste experimento para diferentes arquiteturas são apresentados na Tabela 2.

Tabela 2: Resultados obtidos no experimento 2.

Redes Neurais	Acurácia de Treinamento	Acurácia de Validação	Acurácia de Teste
VGG19	0,9199	0,8925	0,8901
VGG16	0,9169	0,8925	0,8990
Inception-V3	0,6771	0,8665	0,6750
Xception	0,7496	0,9012	0,7394

5.2.1 Experimento 3

No experimento 3, as mesmas técnicas de aumentação de dados do experimento 2 foram aplicadas às imagens da base. Entretanto, utilizou-se a função *ImageDataGenerator* para fazer o pré-processamento das imagens, ou seja, tem-se a normalização das imagens para tornar

os pixels no intervalo de [1-0] e a aplicação da aumentação de dados igual ao que ocorre no experimento 2. Os resultados obtidos neste experimento para diferentes arquiteturas são apresentados na Tabela 3.

Tabela 3: Resultados obtidos no experimento 3.

Redes Neurais	Acurácia de Treinamento	Acurácia de Validação	Ajuste Fino	Acurácia de Teste
VGG19	0,5818	0,7425	acc 0,7155 val_acc 0,8420	0,7018
VGG16	0,6294	0,7970	acc 0,7384 val_acc 0,8720	0,7224
Inception-V3	0,8586	0,9400	Não foi necessário	0,8312
Xception	0,9982	0,9955	Não foi necessário	0,9899

Como neste experimento os valores de acurácia ficaram abaixo de 65% para as arquiteturas VGG-19 e VGG-16, a técnica de ajuste fino foi aplicada para re-treinar os modelos nesses casos.

5.3 Experimento 4

Neste experimento, as técnicas de aumentação de dados foram aplicadas na arquitetura VGG por meio da camada de aumentação de dados, como mostrado na Figura 10, e a função *ImageDataGenerator* foi utilizada para a arquitetura *Inception-V3*, sendo responsável por aplicar as técnicas de aumentação de dados sem requerer uma camada na rede neural. Testes não foram realizados com o modelo *Xception*, pois o valor de acurácia já era elevada. As seguintes estratégias foram aplicadas neste experimento:

- Rotação aleatória: ângulos de até 20°na rotação das imagens.
- Ampliação: aumento de 1.3x sobre o tamanho das imagens.
- Deslocamento horizontal: translação em 10 pixeis.

Os resultados obtidos neste experimento são apresentados nas Tabelas 4 e 5.

Tabela 4: Resultados obtidos no experimento 4 - CNN Inception.

Técnica de Aumentação de Dados	Acurácia de Treinamento	Acurácia de Validação	Acurácia de Teste
Espelhamento Horizontal Aleatório	0,9101	0,9395	0,9071
Espelhamento Vertical Aleatório	0,8130	0,9200	0,8109
Rotação Aleatória	0,8717	0,9370	0,8691
Ampliação Aleatória	0,9107	0,9425	0,9100

Como as arquiteturas VGG-16 e VGG-19 comportaram-se de maneira similar quando comparadas com o uso de técnicas de aumentação de dados, não foi necessário repetir o experimento utilizando a rede VGG-19.

Tabela 5: Resultados obtidos no experimento 4 - CNN VGG16.

Técnica de Aumentação de Dados	Acurácia de Treinamento	Acurácia de Validação	Acurácia de Teste
Espelhamento Horizontal Aleatório	0,9797	0,9050	0,9721
Espelhamento Vertical Aleatório	0,9152	0,8945	0,9091
Rotação Aleatória	0,9462	0,9090	0,9413
Ampliação Aleatória	0,9709	0,9105	0,9623

6 Conclusões

Este trabalho estudou um conjunto de técnicas de aumentação de dados e de transferência de conhecimento, utilizando redes neurais pré-treinadas para a classificação de pássaros. A partir dos experimentos descritos na Seção 5, pode-se observar alguns fatos com as redes do tipo VGG, que têm uma melhor precisão quando uma camada de aumentação de dados é inserida em comparação com as redes Inception-V3 e Xception, que têm melhor precisão quando usada a função de pré-processamento *ImageDataGenerator*.

No experimento 4, utilizando certas técnicas de aumentação de dados, uma acurácia maior foi obtida em comparação com as demais técnicas, podendo-se concluir que, para um conjunto de dados e definida uma rede neural, pode-se determinar uma técnica de aumentação de dados que melhora a acurácia do modelo criado. Em resumo, a partir desses dois itens (conjunto de dados e a rede neural), pode-se determinar uma técnica ou uma combinação de técnicas de aumentação de dados que refinam o modelo.

Para compreender melhor uma possível causa desses fatos, deve-se voltar inicialmente à Seção 2, a qual descreve as arquiteturas das redes. As redes do tipo VGG são baseadas em camadas sequenciais, ou seja, uma rede profunda, enquanto as redes Xception e Inception-V3 utilizam-se da estratégia dos módulos, ou seja, uma estratégia baseada tanto na profundidade quanto na largura da rede. Diante deste fato, pode-se assumir que em redes mais profundas, como a VGG, têm uma melhor acurácia quando se utiliza uma camada de aumentação de dados, enquanto redes que se baseiam em largura, como a Xception e Inception-V3, têm uma melhor acurácia quando utilizada a função de pré-processamento de dados. Assim, ao aplicar as técnicas de aumentação de dados no experimento 4, haveria uma queda na acurácia dos modelos.

Como direções para trabalhos futuros, sugere-se uma investigação minuciosa nas arquiteturas das redes neurais e assim explorar, por meio de mais experimentos, a relação entre o tipo de arquitetura e outras técnicas de aumentação de dados.

Referências

[1] BIRDS 400 - Species Image Classification, Acesso: 2022-06-24. `https://www.kaggle.com/datasets/gpiosenka/100-bird-species`.

[2] Transfer learning and fine-tuning, Acesso: 2022-06-24. `https://www.tensorflow.org/guide/keras/transfer_learning`.

[3] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, and M. Devin. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[4] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, and A. Belopolsky. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, pages arXiv–1605, 2016.

[5] J. R. Almeida. Transfer learning e convolutional neural networks para a classificação de imagens e reconhecimento de objetos no âmbito da perícia criminal. Master's thesis, Instituto de Ciências Exatas, Departamento de Ciência da Computação, Universidade de Brasília, Brasil, 2020.

[6] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[7] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1251–1258, 2017.

[8] F. Chollet. Keras: The Python deep learning library. *Astrophysics source code library*, pages ascl–1806, 2018.

[9] A. C. Ferreira, L. R. Silva, F. Renna, H. B. Brandl, J. P. Renoult, D. R. Farine, R. Covas, and C. Doutrelant. Deep learning-based methods for individual recognition in small birds. *Methods in Ecology and Evolution*, 11(9):1072–1085, 2020.

[10] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.

[11] Y.-P. Huang and H. Basanta. Bird image retrieval and recognition using a deep learning platform. *IEEE Access*, 7:66980–66989, 2019.

[12] S. Kahl, C. M. Wood, M. Eibl, and H. Klinck. BirdNET: A deep learning solution for avian diversity monitoring. *Ecological Informatics*, 61:101236, 2021.

[13] U. M. Kamlesh Borana and V. S. Rajdeep Sodha. Bird Species Identifier using Convolutional Neural Network. *International Journal of Engineering Research & Technology*, 09, 2020.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

[15] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2009.

[16] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, and L. Antiga. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[18] F. Seide and A. Agarwal. CNTK: Microsoft's open-source deep-learning toolkit. In *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2135–2135, 2016.

[19] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, abs/1409.1556, 2015.

[20] J. Song and S. Li. Bird audio detection using convolutional neural networks and binary neural networks. In *Proceedings of DCASE Challenge*, 2018.

[21] D. Stowell and M. D. Plumbley. Automatic large-scale classification of bird sounds is strongly improved by unsupervised feature learning. *PeerJ*, 2:e488, 2014.

[22] D. Stowell, E. Benetos, and L. F. Gill. On-Bird Sound Recordings: Automatic Acoustic Recognition of Activities and Contexts. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(6):1193–1206, 2017.

[23] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[24] J. Xie, K. Hu, M. Zhu, J. Yu, and Q. Zhu. Investigation of different CNN-based models for improved bird sound classification. *IEEE Access*, 7:175353–175361, 2019.

[25] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.