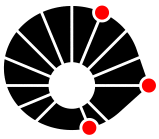
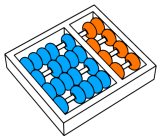




Identificação de Espécies de Árvores por Imagens de Tronco Utilizando Aprendizado de Máquina Profundo

Anderson Nogueira Cotrim *Helio Pedrini*

Relatório Técnico - IC-PFG-19-50
Projeto Final de Graduação
2019 - Dezembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

The contents of this report are the sole responsibility of the authors.
O conteúdo deste relatório é de única responsabilidade dos autores.

Identificação de Espécies de Árvores por Imagens de Tronco Utilizando Aprendizado de Máquina Profundo

Anderson Nogueira Cotrim* Hélio Pedrini†

Resumo

A identificação de espécies de árvores por imagens de tronco é uma tarefa desafiadora e de grande interesse no reconhecimento florestal. Neste trabalho, técnicas de aprendizado de máquina profundo foram exploradas para a resolução deste problema utilizando as bases de dados públicas `BarkNet`, `Trunk12` e `Bark-101`, cujas taxas de acurácia obtidas foram 94,05%, 96,43% e 51,04%, respectivamente, utilizando o protocolo de validação cruzada com 5 partições nas duas primeiras bases e seu próprio conjunto de teste na última base.

1 Introdução

A classificação de espécies de árvores por imagem de tronco é um problema que vem ganhando interesse pela comunidade de visão computacional. A baixa variabilidade interclasse e alta variabilidade intraclasse dessa tarefa torna difícil o desenvolvimento de algoritmos e técnicas que sejam eficazes em termos de acurácia.

Não apenas máquinas, mas mesmo os seres humanos, possuem dificuldade em tarefas dessa natureza. Segundo Fiel e Sablatnig [3], dois especialistas obtiveram 56.6% e 77.6% de acurácia na base de dados *Australian Federal Forest* (AFF), composta por 11 classes e 1183 imagens, porém não disponível publicamente.

A classificação de espécies de árvores é uma tarefa de interesse na aplicação de reconhecimento florestal. Recentemente, a comunidade da área de robótica tem investigado aplicações com *drones* em florestas [11]. Neste contexto, a imagem do tronco das árvores é interessante devido a três fatores: é uma característica invariante às estações do ano (diferentemente de folhas, frutos, flores), altera-se pouco ao longo de vários anos e são mais fáceis de serem coletadas de forma isolada do plano de fundo da imagem (*background*) em comparação a frutas e folhas que podem estar praticamente inacessíveis em árvores altas.

Grande parte dos trabalhos vem formulando o problema de classificação de tronco de árvores como um problema de classificação de textura, baseando-se em métodos tradicionais (*hand-crafted*) para extração de características como filtros de Gabor, *Scale-Invariant Feature Transform* (SIFT), *Local Binary Patterns* (LBP) e variações desses descritores.

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

A maioria das bases de dados disponíveis para esse problema possui poucas imagens. Além disso, poucas são disponíveis publicamente. Recentemente, uma nova base de dados, chamada `BarkNet` [1], tornou possível o estudo da aplicação de técnicas de aprendizado de máquina (*deep learning*) para resolução do problema, devido a sua grande quantidade de imagens. Este é o foco deste trabalho, que explora diversas redes convolucionais que se destacaram por serem estado da arte na classificação de imagens na área de aprendizado de máquina.

Neste trabalho, a base de dados `BarkNet` foi utilizada para explorar arquiteturas tais como `inceptionV3`, `densenet`, `resnet`, `mobilenetV2`, `vgg`, bem como investigar técnicas como aumentação de dados. Ao final deste processo, o melhor modelo obtido foi utilizado para verificar a viabilidade de transferência de aprendizado em bases de dados menores. Em particular, duas bases públicas foram utilizadas: `Trunk12` [12] e `Bark-101` [8].

A Seção 2 descreve alguns dos conceitos utilizados para realização deste trabalho. A Seção 3 apresenta resultados de outras abordagens que utilizaram as mesmas bases de dados avaliadas neste trabalho, ou seja, `BarkNet`, `Trunk12` e `Bark-101`. A Seção 4 descreve a metodologia do trabalho, bem como as bases de dados utilizadas para os experimentos. A Seção 5 apresenta e discute os principais resultados obtidos nos experimentos. Finalmente, a Seção 6 conclui o trabalho e sugere algumas propostas para futuros trabalhos.

2 Conceitos Relacionados

Métodos de aprendizado de máquina profundo (*deep learning*) são atualmente o estado da arte em muitos problemas, em particular, classificação de imagens. Métodos que utilizam aprendizado de máquina (profundo ou raso) buscam encontrar um modelo (por exemplo, regras e parâmetros) utilizando um conjunto de dados (exemplos) e um método para guiar o aprendizado do modelo a partir desses exemplos. Enquanto as técnicas tradicionais de aprendizado de máquina são modelos de funções simples em relação à entrada, as técnicas de aprendizado profundo utilizam modelos de funções compostas em muitas camadas, de forma a criar, em cada nível, representações intermediárias dos dados até a representação que corresponde à saída desejada [10].

Há diversos métodos de aprendizado profundo [4, 6, 9] como *Deep Belief Networks* (DBN), *Deep Boltzmann Machines* (DBM), *Recurent Neural Networks* (RNN), *Generative Adversarial Networks* (GAN), cada uma com foco em solução de um problema específico. Para a classificação supervisionada de imagens, o estado da arte se encontra nas Redes Neurais Convolucionais (*Convolutional Neural Networks* - CNN), o qual é o foco deste trabalho.

Redes Neurais Convolucionais são conhecidas por serem compostas basicamente de camadas convolucionais. A convolução da entrada por uma matriz de pesos, que fazem parte dos parâmetros a serem aprendidos na etapa de treinamento, pode ser interpretada como uma operação de filtragem da entrada, de forma a realçar características importantes para uma predição correta [5]. A Figura 1 apresenta o resultado da convolução da imagem pelo filtro de Sobel, muito utilizado para realçar linhas verticais ou horizontais de uma imagem.

Em uma rede neural convolucional, as operações de convolução são aplicadas em cascata,

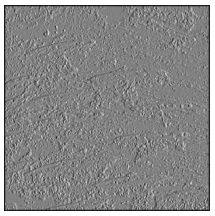
-1	0	+1
-2	0	+2
-1	0	+1

(a) Sobel X

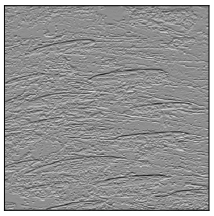
+1	+2	+1
0	0	0
-1	-2	-1

(b) Sobel Y

(c) Entrada



(d) Resultado após filtragem com operador Sobel X



(e) Resultado após filtragem com operador Sobel Y

Figura 1: Exemplo de uma aplicação de filtros. As Figuras 1(a) e 1(b) apresentam as matrizes com pesos conhecidos como filtro de Sobel X e Sobel Y, respectivamente. O primeiro visa realçar linhas verticais de uma imagem, enquanto o segundo visa realçar linhas horizontais. A Figura 1(d) é o resultado após a aplicação do filtro de Sobel X (Figura 1(a)) no exemplo de entrada (Figura 1(c)). Similarmente, a Figura 1(e) é o resultado após a aplicação do filtro de Sobel Y (Figura 1(b)) no mesmo exemplo de entrada (Figura 1(c)). Pode-se observar que o filtro de Sobel Y apresenta um resultado mais destacável devido à predominância de linhas horizontais na imagem de entrada.

em conjunto com outras operações auxiliares como *batch normalization*, *activation functions*, *pooling* e, ao final, geralmente há um classificador profundo do tipo *Mutilayer Perceptron*, o qual também possui parâmetros a serem aprendidos, de forma que o último conjunto de representações feita pelas convoluções produza a saída esperada [10].

O processo de treinamento de uma rede convolucional consiste em ajustar todos os seus parâmetros de forma que uma função de custo seja minimizada. A função de custo essencialmente depende da saída da rede e a saída esperada. Podemos interpretá-la como uma função que calcula o quanto a saída da rede está errada em relação a saída esperada. Dentro de várias funções de custo, uma das mais utilizadas é a função de entropia cruzada [5]. A minimização da função de custo ocorre por diversas iterações sobre a rede. Para cada iteração, calculam-se os custos para um pequeno conjunto de imagens (*batch*) e os parâmetros da rede são re-computados por meio de um processo conhecido na literatura como retropropagação de erros dependente da taxa de atualização (*learning rate*), que é um hiperparâmetro definido antes do treinamento [4]. A Figura 2 ilustra o funcionamento geral para treinamento de uma rede convolucional.

As primeiras camadas tendem a gerar representações mais simples ou genéricas (como exemplo, o filtro de Sobel), que não estão diretamente relacionadas à resolução do problema em questão, entretanto, a combinação dessas representações por meio da convolução com outras tende a gerar representações cada vez mais complexas e, portanto, cada vez mais especializadas ao problema à medida que avançamos nas camadas de uma rede. Isso nos permite utilizar técnicas como transferência de aprendizagem, que consiste em treinar uma rede para uma tarefa e retreiná-la para outra, a qual normalmente possui uma quantidade menor de dados, mantendo-se as primeiras camadas, já que essas geram representações mais genéricas. O número de camadas a serem congeladas para um melhor desempenho dessa técnica essencialmente depende da semelhança entre os problemas [5].

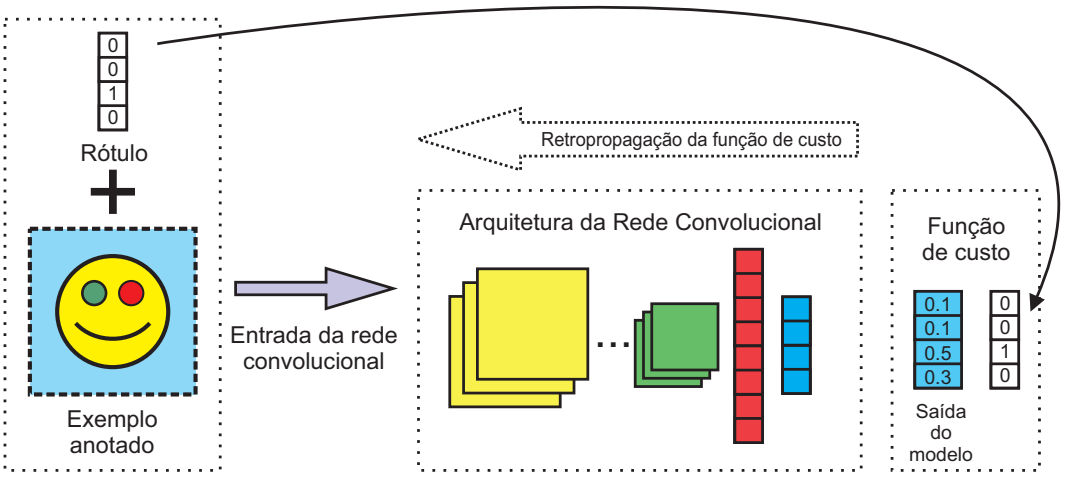


Figura 2: Diagrama ilustrativo do processo de treinamento de uma rede convolucional. O exemplo anotado é composto da imagem e o rótulo de sua classe. A imagem é usada como entrada para a rede, que tem como saída um vetor com as probabilidades de cada classe contida no domínio do problema. O custo é calculado em relação à saída obtida e esperada, e então os pesos são corrigidos da última camada até a primeira em um processo conhecido como retropropagação [10].

Overfitting é um problema bastante recorrente em arquiteturas de redes profundas, que ocorre quando o modelo notavelmente tem um desempenho melhor nos dados utilizados para treinamento do que para validação. Isso ocorre porque os parâmetros da rede foram excessivamente ajustados ao treinamento, o que causa alguns erros no conjunto de validação. Como esse problema é comum, há diversas formas consolidadas na literatura de tentar evitá-lo, como regularização L2, adição de camada *dropout* e aumentação de dados. Este último tem por objetivo aumentar artificialmente a quantidade de dados disponíveis para melhorar a robustez do modelo a novas entradas, por meio da adição de pequenos ruídos, distorções nas cores e escala, que são razoavelmente diferentes para cada tipo de câmera utilizada [4].

O número de camadas, os tipos delas e o arranjo entre as camadas definem o que denominamos arquitetura de uma rede convolucional. Com os desafios da base de dados ImageNet [2], pesquisadores têm proposto constantemente novas arquiteturas de rede neural. Algumas delas ficaram bastante conhecidas por resultarem em boa eficácia, tais como AlexNet em 2012, Google Inception em 2014 e ResNet em 2015. Neste trabalho, aproveitaremos algumas dessas arquiteturas pré-treinadas com a ImageNet na tentativa de adaptá-las à solução de classificação de troncos de árvores.

3 Trabalhos Relacionados

Grande parte dos trabalhos relacionados ao problema, devido à escassez de dados, baseiam-se em abordagens tradicionais de extração de características (*hand-crafted*), que obtêm aproximadamente 70% de acurácia na base de dados AFF [1]. Ratajczak et al. [8] obtiveram,

nesta categoria de abordagem, uma acurácia de 80.7% de acurácia na base AFF, utilizando o protocolo *Leave-One-Out* (LOO), sendo um resultado satisfatório quando comparado à acurácia de humanos especialistas [3] para a mesma tarefa.

Ratajczak et al. [8] efetuaram uma compilação de resultados obtidos pelas abordagens dos autores em várias bases, além da AFF. O melhor resultado desse trabalho para a base de dados `Trunk12` foi de 84.2% em acurácia com o mesmo protocolo LOO. Além deste resultado, o trabalho propôs uma base de dados mais desafiadora do que as que existiam até então em termos de ruídos, variações de parâmetros da câmera e número de classes. Esta base, chamada `Bark-101`, é composta de 101 classes de espécies de árvores. Como esperado, os métodos desenvolvidos no trabalho tiveram uma eficácia pior para esta base, sendo o melhor resultado uma acurácia de 41.9%, em uma abordagem em que 50% dos dados foram utilizados para treinamento e 50% para teste.

Graças à base `BarkNet` [1], tornou-se possível a investigação do uso de redes profundas para resolver o problema da classificação de árvores por imagens do tronco. O próprio autor da base de dados conseguiu obter uma acurácia de 93.88% por imagem utilizando o protocolo de validação cruzada com 5 partições. Além disso, o trabalho apresentou conclusões interessantes a respeito da utilização de redes profundas para essa tarefa. Por exemplo, obter imagens de uma grande variedade de árvores é mais relevante na etapa de treinamento do que utilizar a variedade de imagens de uma mesma árvore (fotografias de diferentes partes do tronco), muito embora este último também aumente a eficácia. No entanto, isto é mais relevante na etapa de predição, em que se obteve uma acurácia de 97.81% (utilizando todas as imagens de um mesmo tronco).

4 Materiais e Métodos

Nesta seção, descrevemos os materiais utilizados para os experimentos (Seção 4.1) e a metodologia para desenvolvida neste trabalho (Seção 4.2).

4.1 Bases de Dados

A Tabela 1 sumariza as principais características das bases de dados utilizadas neste trabalho e os principais resultados obtidos pelos trabalhos [1] e [8]. Pode-se observar na tabela o quanto a `BarkNet` possui um número maior de dados em relação às outras bases. A Figura 3 mostra um exemplo de cada uma das 23 classes da base de dados `BarkNet`.

A Tabela 2 sumariza as classes contidas na base de dados `BarkNet`, com o número de árvores e imagens. Observa-se que três classes (ERB, PEG e PID) possuem um número extremamente reduzido de árvores. Por essa razão e pela abordagem de aprendizado profundo essas classes foram excluídas de qualquer experimento neste trabalho, restando apenas 20 classes utilizáveis.

Conforme mencionado, o problema de classificação de árvores pela imagem do tronco é desafiador devido à alta variabilidade intraclasse e a alta semelhança interclasse. A Figura 4 mostra esse fato por meio de exemplos de duas classes visualmente semelhantes, ORA e ERR. Cada uma delas possui imagens um pouco destoantes dentro de uma mesma classe.

Tabela 1: Características das bases de dados utilizadas neste trabalho.

Características	**BarkNet**	**Trunk12**	**Bark-101**
Número de classes	23	12	101
Número total de imagens	23616	393	20592
Dimensão das imagens	(260-2984)×(370-5312)	1000×1334	(69-800)×(112-804)
Espaço em disco (arquivos JEPG)	32.5 GB	1.1 GB	387.6 MB
Variação na iluminação	✓	✗	✓
Variação na escala	✓	✗	✓
Ruído	✓	✗	✓
Acurácia *top-1* de alguns trabalhos	93.88% (validação cruzada) [1]	84.2% (LOO) [8]	41.9% (50/50) [8]

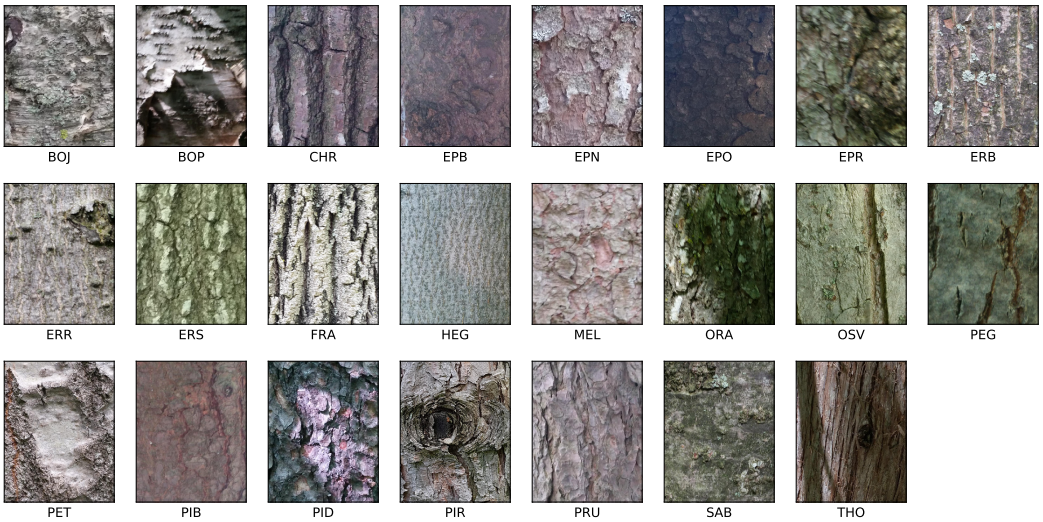


Figura 3: Exemplos de imagens para cada uma das 23 espécies contidas na base `BarkNet`.

(a) Exemplos de imagens da classe ERR (b) Exemplos de imagens da classe ORA

Figura 4: Exemplos de imagens que possuem alta variabilidade intraclasse e baixa variabilidade interclasse (classes bem semelhantes). Pode-se observar, a partir da cor das imagens, que algumas possuem uma tonalidade mais verde causada pelos ajustes impróprios de balanço de cor feito pelas câmeras.

Segundo Carpentier et al. [1], as imagens para composição da `BarkNet` foram coletadas

Tabela 2: Composição da base de dados `BarkNet`.

Código	**Nome Científico**	**Nome Comum**	**N. Árvores**	**N. de Imagens**
BOJ	*Betula alleghaniensis*	Yellow birch	43	1255
BOP	*Betula papyrifera*	White birch	32	1285
CHR	*Quercus rubra*	Northern red oak	109	2724
EPB	*Picea glauca*	White spruce	44	596
EPN	*Picea mariana*	Black spruce	44	885
EPO	*Picea abies*	Norway spruce	72	1324
EPR	*Picea rubens*	Red spruce	27	740
ERB	*Acer platanoides*	Norway maple	1	70
ERR	*Acer rubrum*	Red maple	64	1676
ERS	*Acer saccharum*	Sugar maple	81	1999
FRA	*Fraxinus americana*	White ash	61	1472
HEG	*Fagus grandifolia*	American beech	41	840
MEL	*Larix laricina*	Tamarack	77	1874
ORA	*Ulmus americana*	American elm	25	767
OSV	*Ostrya virginiana*	American hophornbeam	29	612
PEG	*Populus grandidentata*	Big-tooth aspen	3	64
PET	*Populus tremuloides*	Quaking aspen	58	1037
PIB	*Pinus strobus*	Eastern white pine	39	1023
PID	*Pinus rigida*	Pitch pine	4	123
PIR	*Pinus resinosa*	Red pine	29	596
PRU	*Tsuga canadensis*	Eastern Hemlock	45	986
SAB	*Abies balsamea*	Balsam fir	41	922
THO	*Thuja occidentalis*	Northern white cedar	38	746

buscando variação em iluminação, escala e ruído, visando a um cenário mais realista. Além disso, com intuito de aumentar a robustez da base, os autores utilizaram quatro modelos de câmeras, conforme a distribuição ilustrada na Figura 5.

4.2 Metodologia

Este trabalho possui uma metodologia exploratória das técnicas mais utilizadas de classificação de imagens por redes convolucionais. Por meio do aprendizado supervisionado, almejou-se calibrar parâmetros de forma que o modelo consiga generalizar novas amostras com boa acurácia.

Para treinamento dos modelos, explorou-se técnicas como aumentação de dados, transferência de aprendizagem com pesos da `ImageNet` e o parâmetro de regularização L2. Como função de custo, utilizou-se a entropia cruzada, cujo erro foi retropropagado sob regras do otimizador *Adam*. A Figura 6 ilustra o funcionamento geral da metodologia de classificação de espécies de árvores proposta neste trabalho.

Além disso, para evitar o treinamento de um classificador com viés às classes que contêm mais exemplos, optou-se por fazer uma seleção aleatória estratificada em relação às classes e às árvores, de forma que cada época de treinamento seja equilibrada em termos de número de exemplos em cada classe.

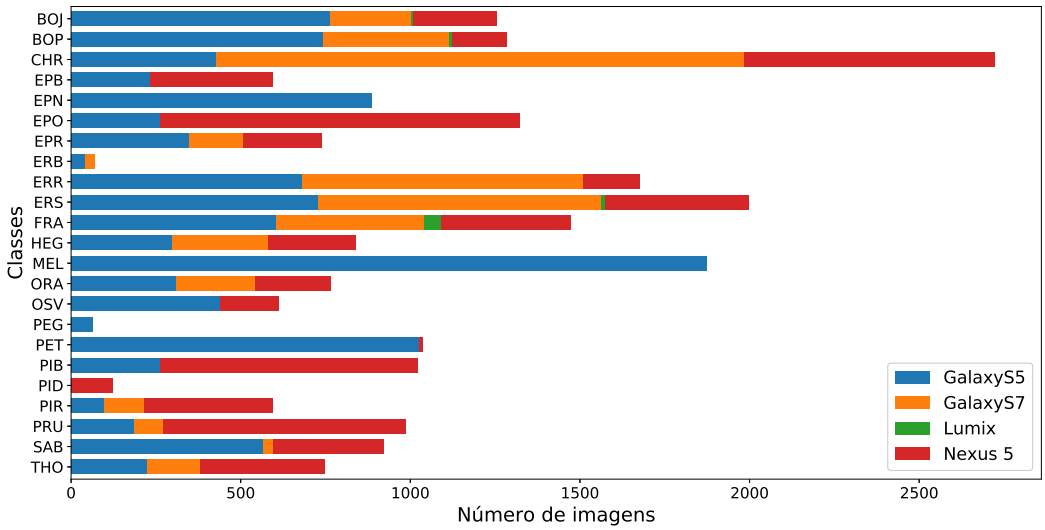


Figura 5: Distribuição da base em relação ao tipo de câmera.

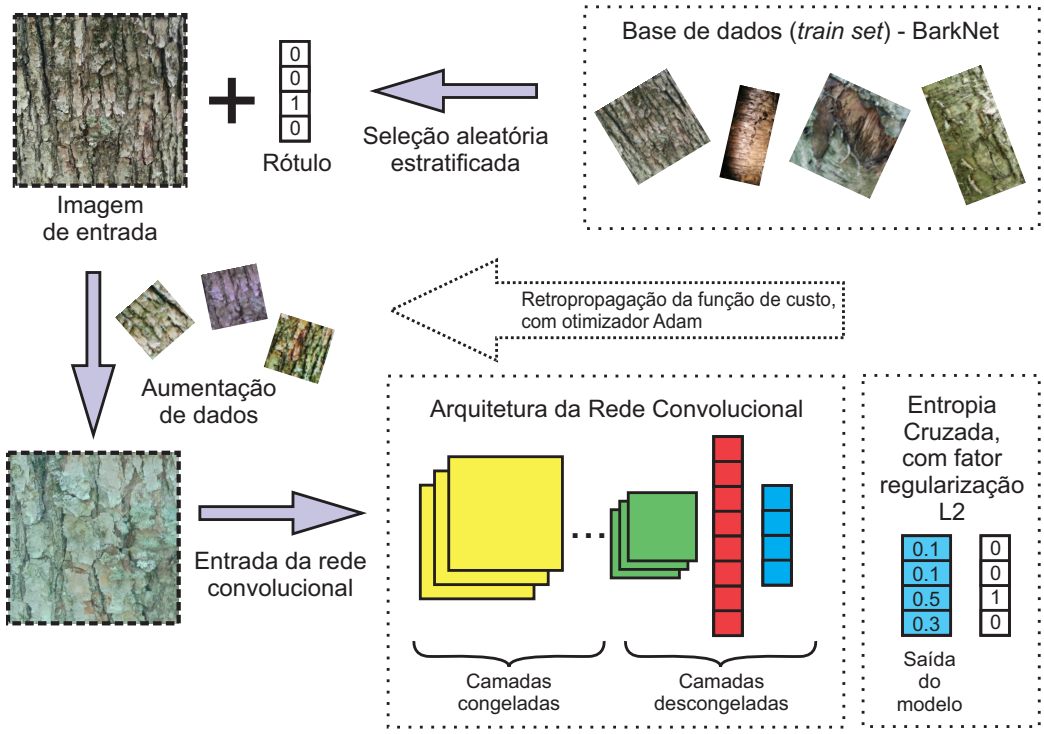


Figura 6: Diagrama ilustrativo da abordagem de classificação de espécies de árvores por redes convolucionais.

4.3 Métricas de Avaliação

Para avaliação dos experimentos, as bases de dados foram divididas, estratificadamente, em conjuntos de treinamento, validação e teste de forma única, utilizada igualmente em todos os

experimentos. No caso da `BarkNet`, como foram explorados diversos métodos e arquiteturas de redes convolucionais, optou-se por uma divisão completa: 64% para treinamento, 16% para validação e 20% para teste. Além disso, para experimentos que exigiram uma análise mais acurada, utilizamos uma divisão de 5 partições nesta base. No caso da base `Trunk12`, como não houve muitos experimentos, optou-se por utilizar 5 partições, sem manter um conjunto de testes. A base `Bark-101` possui conjuntos próprios de treinamento/validação, os quais estão na proporção 50%/50%.

As avaliações foram baseadas na entropia cruzada média obtida para cada conjunto. A avaliação da acurácia *top-1* de cada conjunto de dados pode ser realizada de duas maneiras: predição de um recorte aleatório da imagem (*single crop*) ou por diversos recortes da imagem (*multiple crop*) para que a imagem inteira contribua para o resultado (utilizando a predição mais recorrente).

Imagens com dimensões menores do que as dimensões originais das arquiteturas experimentadas (299 para `InceptionV3` e 224 para as outras) foram completadas com *padding* reflexivo, o que é razoável dada a natureza de texturas das imagens. No caso da predição *multiple crop*, o tamanho da imagem teve de ser redimensionado para o próximo inteiro divisível em relação à dimensão de entrada da rede, o que possibilitou o uso de toda a imagem para predição.

5 Experimentos e Resultados

Esta seção descreve as condições tanto em *hardware* quanto em *software* para realização dos experimentos (Seção 5.1), bem como a forma como foram realizados experimentos para geração de resultados em cada etapa (Seção 5.2).

5.1 *Setup* Experimental

Todos os experimentos deste trabalho foram realizados em uma máquina com processador i5 9600k, 16GB de memória e placa gráfica RTX 2070 8 GB sob sistema Linux Ubuntu 18.04. O driver da placa gráfica utilizada se encontra na versão 430.50, com CUDA 10.0 e cuDNN v7.6.3. A linguagem de programação utilizada foi Python 3.6.8 com a biblioteca PyTorch [7] na versão 1.2.0.

5.2 Procedimento Experimental e Resultados

Dividimos os experimentos realizados neste trabalho em quatro etapas principais que foram executadas sequencialmente no intuito de se obter resultados satisfatórios em cada uma delas. A primeira tem por objetivo encontrar o melhor redimensionamento para imagens da base `BarkNet` para que se obtenha o melhor resultado de uma rede profunda. As próximas etapas utilizam este redimensionamento. A segunda etapa tem por objetivo explorar a aumentação de dados para melhorar o resultado dos modelos ou os deixarem mais robustos a novos tipos de dados. A terceira etapa tem por objetivo explorar as diferentes arquiteturas de redes profundas do estado da arte com treinamento que foi feito utilizando a aumentação de dados obtida na segunda etapa. Por fim, na quarta etapa, selecionamos o melhor modelo

da terceira etapa e exploramos técnicas de transferência de aprendizagem para as bases com menor quantidade de dados: `Trunk12` e `Bark-101`.

As próximas subseções detalham os procedimentos adotados e os resultados obtidos em cada uma das etapas.

5.2.1 Encontrando Melhor Redimensionamento de Imagem para `BarkNet`

Esta etapa experimental consistiu em encontrar o melhor fator de redução da imagem para um bom desempenho de classificadores por redes convolucionais na `BarkNet`. Para isso, adotou-se a rede `resnet18`, a mesma utilizada por Carpentier et al. [1] e sob as mesmas condições de treinamento: 40 épocas, *learning rate* inicial de 0.0001 com redução de fator 5 nas épocas 16 e 33 e regularização L2 de 0.0001. Este procedimento foi realizado para fatores de redução nas duas dimensões das imagens por 1.00 (imagem com dimensões inalteradas), 0.67, 0.50, 0.40, 0.33, 0.29 e 0.25, resultados que podem ser visualizados na Figura 7.

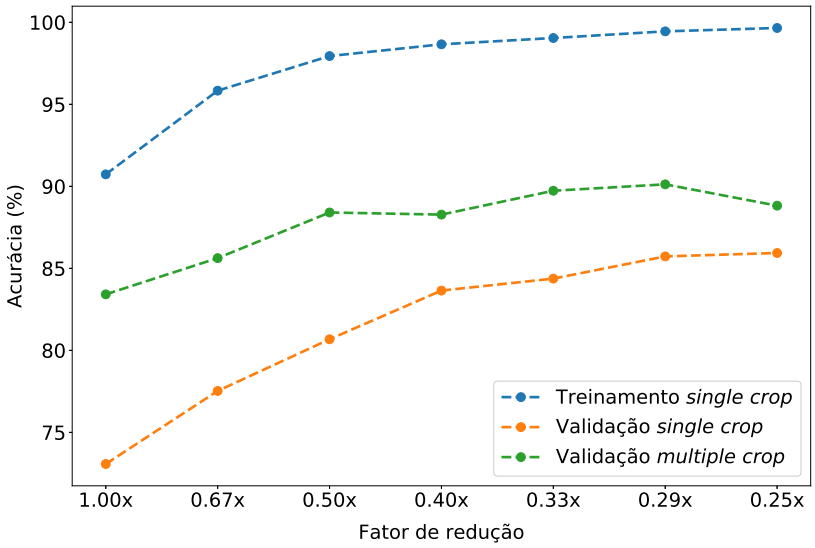


Figura 7: Gráfico da acurácia obtida para alguns valores de redimensionamento das imagens.

Conforme pode ser visualizado na Figura 7, há uma certa estabilização no resultado da acurácia na validação *multiple crop* para fatores de redução menores ou iguais a 0.5, enquanto a acurácia no treinamento e no caso *single crop* tende a aumentar à medida que reduzimos mais a imagem - resultado completamente compreendido pelo fato do treinamento e validação *single crop* utilizarem apenas uma parte da imagem, enquanto *multiple crop* utiliza a imagem completa para gerar o resultado.

À medida que diminuímos a imagem, estamos aproximando a validação *single crop* do *multiple crop*, pois *single crop* tende cada vez mais utilizar uma parte maior da imagem. Em contrapartida, cada vez mais estamos perdendo detalhes menores da imagem e reduzindo o número de dados da base de dados, o que gera a tendência de aumento da acurácia também no treinamento, piorando o *overfitting*. Por esses motivos, escolhemos adotar o fator de redução 0.5 como padrão para todos os próximos experimentos, não reduzindo bruscamente

os dados dos experimentos e mantendo uma boa acurácia no caso *multiple crop*, que é mais relevante em resultados *single crop*.

5.2.2 Aumentação de Dados

Esta etapa consistiu em explorar técnicas básicas de aumentação de dados com o intuito de diminuir o *overfitting* e, consequentemente, melhorar o resultado na validação. Foram exploradas individualmente as seguintes estratégias:

1. mudanças de escala e razão de aspecto (*aspect ratio*) escolhidos aleatoriamente uniformemente entre 0.8, 1.2 e 0.75, 1.25, respectivamente.
2. pertubação de brilho, contraste e saturação de uma imagem por fatores escolhidos aleatoriamente, uniformemente entre 0.8 e 1.2.
3. espelhamento (*flip*) vertical com probabilidade 0.5.
4. espelhamento (*flip*) horizontal com probabilidade 0.5.

A Tabela 3 sumariza os resultados para cada uma dos testes, conforme as técnicas avaliadas.

Tabela 3: Experimentos de aumentação de dados na `BarkNet`. Todos utilizaram a arquitetura `resnet18` pré-treinada na `ImageNet` (com primeira camada congelada) treinada com 40 épocas, com *learning rate* de 0.0001 com redução de fator 5 nas épocas 16 e 33 e regularização L2 de 0.0001.

	Acurácia (%)				
	Hold-out			**Validação Cruzada**	
Estratégia	**Treinamento** *single*	**Validação** *single*	**Validação** *multiple*	**Validação** *single*	**Validação** *multiple*
Sem aumentação	96.84	82.04	87.08	84.06	90.42
Escala e razão de aspecto	96.82	80.89	86.25	83.55	90.20
Color Jitter	96.93	81.23	86.43	83.98	89.98
Espelhamento vertical	96.71	80.84	87.54	83.69	89.49
Espelhamento horizontal	97.07	82.08	87.73	83.90	89.78

Conforme pode ser observado na Tabela 3, nenhum dos experimentos trouxe resultados significativos. O espelhamento horizontal aparenta ser a técnica mais confiável para aumentar os dados, o espelhamento vertical assumiria uma possível captura do tronco com a câmera em uma posição invertida, enquanto as outras técnicas introduzem valores de pixels artificiais, o que provavelmente traz mais ruídos ao modelo do que uma melhoria na capacidade de generalização para novas amostras. Por esses motivos, adotaremos o espelhamento horizontal nos próximos experimentos.

Para uma verificação mais detalhada das técnicas no sentido de melhorar a generalização do modelo, repetiu-se o mesmo experimento utilizando todos os dados da Câmera Nexus 5

como validação e o restante foi considerado parte do conjunto de treinamento. Entretanto, conforme pode ser observado na Figura 5, algumas classes contêm poucas imagens da Nexus 5, por isso, as classes ERB, PEG, PID, PET, PEG, MEL e EPN foram desconsideradas neste experimento. A Tabela 4 apresenta os resultados obtidos.

Tabela 4: Experimento de aumentação de dados na base `BarkNet` utilizando todas as imagens capturadas com a câmera Nexus 5 para teste. Todos utilizaram a arquitetura `resnet18` pré-treinada na `ImageNet` (com primeira camada congelada) treinada com 40 épocas, com otimizador `Adam` com *learning rate* de 0.0001 com redução de fator 5 nas épocas 16 e 33 e regularização L2 de 0.0001.

	Acurácia (%) – Hold-out		
Estratégia	**Treinamento** *single*	**Validação** *single*	**Validação** *multiple*
Sem aumentação	96.33	67.91	75.07
Escala e razão de aspecto	96.03	66.44	71.67
Color Jitter	96.01	65.56	71.99
Espelhamento vertical	96.14	67.23	72.36
Espelhamento horizontal	96.02	66.09	72.58

Conforme pode-se analisar na Tabela 3, as conclusões são similares na avaliação *hold-out* e validação cruzada com 5 partições. Por isso, optou-se na Tabela 4 por não utilizar o protocolo validação cruzada para permitir realização mais rápida de experimentos. Conforme constatado na Tabela 4, não houve melhora significativa que justificasse a necessidade de uma investigação mais profunda dessas técnicas de aumentação de dados.

5.2.3 Explorando Diversas Redes

Nesta etapa, exploramos outras arquiteturas de redes neurais. Para isso, tivemos que ajustar alguns hiperparâmetros de cada uma das redes para obtenção de melhores resultados. A Tabela 5 resume os resultados das avaliações com *hold-out* das arquiteturas experimentadas.

Conforme pode ser analisado na Tabela 5, os resultados entre as melhores arquiteturas tendem a apresentar valores próximos, em especial, `wide_resnet101_2`, `vgg19_bn`, `resnext50_32x4d` e `densenet201` apresentaram os melhores valores em um dos protocolos avaliados. Portanto, estas são arquiteturas que merecem uma investigação mais detalhada utilizando o protocolo de validação cruzada com 5 partições. O resultado dessa investigação se encontra na Tabela 6.

Conforme constatado na Tabela 6, a melhor arquitetura foi a `densenet201`, ligeiramente melhor do que a `resnext50_32x4d`. A Tabela 7 mostra testes dessa arquitetura para diversos níveis de congelamento das camadas com os pesos da `ImageNet`.

Pode-se notar que o melhor resultado obtido na Tabela 7 foi realizar nenhum congelamento de camadas. Isso pode ser explicado pela diferença entre as tarefas da `ImageNet` e a classificação de troncos, bem como a base de dados `BarkNet` ser suficientemente grande para uma boa convergência do modelo treinando a rede inteira.

Tabela 5: Experimentos em diversas arquiteturas profundas. Todos utilizaram pesos da `ImageNet`, com todas camadas descongeladas, com otimizador `Adam` e sem regularização L2. A tabela mostra o número de épocas, *learning rate* e decaimento adaptados a cada uma para uma melhor convergência possível. Os melhores valores de cada métrica estão destacados em negrito.

	Acurácia (%) – Hold-out			**Perda – Hold-out**	
Arquitetura	**Treinamento** *single*	**Teste** *single*	**Teste** *multiple*	**Teste** *single*	**Teste** *multiple*
alexnet	90.77	73.11	81.82	1.0464	1.0601
densenet121	98.14	86.78	92.74	0.5934	0.5508
densenet201	98.45	87.85	92.96	0.5496	**0.5358**
googlenet	97.24	85.15	91.89	0.6848	0.6414
inception_v3	98.80	85.44	89.99	0.6438	1.1887
resnet18	97.32	86.04	91.15	0.6139	0.5872
resnet152	97.98	86.31	91.11	0.5991	0.6066
resnext50_32x4d	98.54	88.07	91.76	**0.5251**	0.5768
resnext101_32x8d	97.84	88.12	91.04	0.5547	0.6634
vgg11	97.40	83.29	90.33	0.8352	0.7571
vgg19	96.55	83.18	91.06	0.8005	0.7174
vgg19_bn	96.92	86.98	**93.07**	0.6238	0.5824
wide_resnet50_2	98.32	86.71	91.60	0.5902	0.6145
wide_resnet101_2	97.34	**88.32**	91.13	0.5718	0.8120

Tabela 6: Repetição do experimento para algumas arquiteturas que apresentaram bons resultados na Tabela 5 com protocolo de validação cruzada com 5 partições. Os melhores valores de cada métrica estão destacados em negrito.

	Acurácia (%) – 5-fold			**Perda – 5-fold**	
Arquitetura	**Treinamento** *single*	**Teste** *single*	**Teste** *multiple*	**Teste** *single*	**Teste** *multiple*
`densenet201`	98.39	**87.13**	**92.79**	**0.5318**	**0.5300**
`resnext50_32x4d`	98.46	86.84	91.76	0.5550	0.5497
`resnext101_32x8d`	97.62	87.03	91.50	0.5659	0.6318
`vgg19_bn`	96.78	86.35	92.70	0.5969	0.6022
`wide_resnet101_2`	97.08	84.53	89.05	1.0552	2.8361

Conforme pode ser analisado nestes resultados, temos um *overfitting* em todos os experimentos. Como tentativa de reduzir esse efeito, exploramos a regularização L2 para diversos valores em potência de 10, resultado que pode ser analisado na Tabela 8.

A partir da Tabela 8, elegemos o melhor fator de regularização como 0.001. Apesar do valor em acurácia *single* ser ligeiramente melhor em 0.0001, o fator 0.001 trouxe valores para a função de perda bem melhores quando comparados aos demais.

Tabela 7: Experimentos com a `densenet201`, congelando diversos níveis de camadas da arquitetura. Os melhores valores de cada métrica estão destacados em negrito.

	Acurácia (%) – Hold-out			**Perda – Hold-out**	
Camadas congeladas	**Treinamento** *single*	**Teste** *single*	**Teste** *multiple*	**Teste** *single*	**Teste** *multiple*
`denseblock[1..4]`	Perda – 5	68.94	78.61	1.0679	1.2427
`denseblock[1..3]`	97.13	83.16	90.82	0.7557	0.7801
`denseblock[1..2]`	98.39	86.82	92.23	0.5812	0.5970
`denseblock[1]`	98.69	87.85	92.69	0.5724	0.5538
nenhuma	98.45	**87.85**	**92.96**	**0.5496**	**0.5358**

Tabela 8: Experimentos com a `densenet201`, variando o parâmetro de regularização L2. Os melhores valores de cada métrica estão destacados em negrito.

	Acurácia (%) – Hold-out			**Perda – Hold-out**	
Regularização L2	**Treinamento** *single*	**Teste** *single*	**Teste** *multiple*	**Teste** *single*	**Teste** *multiple*
1	29.64	26.76	27.59	2.5068	2.4992
0.1	87.15	77.78	85.84	0.8115	0.8593
0.01	95.86	85.44	90.77	0.5456	0.5635
0.001	98.16	88.23	**93.30**	**0.4885**	**0.4832**
0.0001	98.21	**88.72**	93.16	0.5362	0.5352
0.00001	98.20	88.50	92.90	0.5207	0.5249
nenhuma	98.45	87.85	92.96	0.5496	0.5358

Além dessas arquiteturas mais densas consideradas, resolvemos explorar arquiteturas conhecidas como mais leves, normalmente apropriadas para dispositivos móveis. A Tabela 5 resume os resultados das avaliações com protocolo *hold-out* das arquiteturas leves experimentadas.

Conforme pode ser analisado na Tabela 9, os melhores resultados ficaram entre *mobilenet_v2* e *mnasnet1_0*. Essas arquiteturas foram investigadas de forma mais detalhada utilizando o protocolo de validação cruzada, cujos resultados se encontram na Tabela 10.

Conforme constatado na Tabela 10, a arquitetura `mobilenet_v2` perdeu levemente apenas no quesito de acurácia *single*, a perda da `mobilenet_v2` ficou razoavelmente menor do que a `mnasnet1_0`, portanto, `mobilenet_v2` foi a rede escolhida para os próximos experimentos.

As Tabelas 11 e 12 mostram resultados dos mesmos experimentos feitos com a `densenet201` na `mobilenet_v2`. Sob as mesmas observações feitas para a `densenet201`, a `mobilenet_v2` apresentou um resultado melhor, com um treinamento completo (sem congelar camadas) e com fator de regularização L2 igual a 0.001.

Por fim, utilizando toda base disponível da `BarkNet`, re-treinamos a `densenet201` e a `mobilenet_v2`, com protocolo de validação cruzada com 5 partições, resultado que está

Tabela 9: Experimento em diversas arquiteturas leves. Todas utilizaram pesos da `ImageNet`, todas camadas descongeladas, com otimizador `Adam` e sem regularização L2. Também são apresentados o número de épocas, *learning rate* e o decaimento adaptados a cada uma para uma melhor convergência.

	Acurácia (%) – Hold-out			**Perda – Hold-out**	
Arquitetura	**Treinamento** *single*	**Teste** *single*	**Teste** *multiple*	**Teste** *single*	**Teste** *multiple*
`mnasnet0_5`	93.34	79.59	85.08	0.8288	0.9069
`mnasnet1_0`	98.20	**86.13**	**89.03**	0.7132	0.8303
`mobilenet_v2`	97.11	85.24	88.61	**0.6782**	**0.7416**
`shufflenet_v2_x0_5`	94.90	80.53	83.43	0.8471	0.9657
`shufflenet_v2_x1_0`	95.41	82.51	87.29	0.8287	0.9052
`squeezenet1_0`	87.77	73.98	84.55	1.0366	1.0043
`squeezenet1_1`	87.29	72.58	83.56	1.0683	1.0454

Tabela 10: Repetição do experimento para as arquiteturas `mobilenet_v2`, `mnasnet1_0` da Tabela 9 com protocolo de validação cruzada. Os melhores valores de cada métrica estão destacados em negrito.

	Acurácia (%) – 5-fold			**Perda – 5-fold**	
Arquitetura	**Treinamento** *single*	**Teste** *single*	**Teste** *multiple*	**Teste** *single*	**Teste** *multiple*
`mnasnet1_0`	98.13	**85.45**	89.09	0.6100	0.6809
`mobilenet_v2`	96.81	85.06	**89.12**	**0.5721**	**0.6687**

Tabela 11: Experimentos com a `mobilenet_v2`, congelando diversos níveis de camadas da arquitetura. Os melhores valores de cada métrica estão destacados em negrito.

	Acurácia (%) – Hold-out			**Perda – Hold-out**	
Camadas congeladas	**Treinamento** *single*	**Teste** *single*	**Teste** *multiple*	**Teste** *single*	**Teste** *multiple*
`features[0..18]`	71.76	62.93	74.16	1.2334	1.4319
`features[0..9]`	95.82	83.85	84.63	0.6939	0.8969
`features[0..5]`	96.13	85.23	84.50	0.6772	0.9231
`features[0..2]`	96.40	84.81	88.23	0.6786	0.7784
`features[1]`	96.37	84.37	**88.68**	0.7439	0.7536
nenhuma	97.11	**85.24**	88.61	**0.6782**	**0.7416**

exposto na Tabela 13.

Por fim, a Figura 8 mostra a acurácia por classe dos dois melhores modelos obtidos. Em muitas classes, conforme pode ser observado, há uma diferença bem baixa entre `mobilenet_v2` e `densenet201`. Todas seguem o mesmo padrão, com exceção da classe OSV

Tabela 12: Experimentos com a *mobilenet_v2*, variando o parâmetro de regularização L2. Os melhores valores de cada métrica estão destacados em negrito.

	Acurácia (%) – Hold-out			**Perda – Hold-out**	
Regularização L2	**Treinamento** *single*	**Teste** *single*	**Teste** *multiple*	**Teste** *single*	**Teste** *multiple*
1	5.21	3.14	3.14	2.9958	2.9958
0.1	67.60	57.28	63.36	1.8740	1.9460
0.01	90.35	78.49	86.15	0.7503	0.8166
0.001	95.73	84.66	**89.64**	**0.5772**	**0.6364**
0.0001	96.48	84.66	89.06	0.6417	0.7107
0.00001	96.29	84.81	88.05	0.6482	0.7774
nenhuma	97.11	**85.24**	88.61	0.6782	0.7416

Tabela 13: Avaliação com protocolo de validação cruzada com 5 partições utilizando toda a base de dados `BarkNet`. Todos experimentos utilizaram pesos da `ImageNet`, sem congelamento de camadas, regularização L2 de 0.001 e espelhamento horizontal como aumentação de dados. Na `densenet201`, foram 45 épocas com taxa de aprendizagem inicial de 0.0001 e decaimento por 5 nas épocas 17, 28 e 35. Na `mobilenet_v2`, foram 40 épocas com taxa de aprendizagem inicial de 0.001 e decaimento por 5 nas épocas 15, 25 e 34.

	Acurácia (%) – 5-fold			**Perda – 5-fold**	
Arquitetura	**Treinamento** *single*	**Teste** *single*	**Teste** *multiple*	**Teste** *single*	**Teste** *multiple*
`densenet201`	97.66	88.72	94.05	0.3914	0.4104
`mobilenet_v2`	95.06	85.23	91.47	0.4874	0.5619

que teve um desempenho relativamente pior na `mobilenet_v2` do que a `densenet201`.

Para a `mobilenet_v2`, tivemos o pior desempenho na classe OSV, enquanto na `densenet201` o pior resultado ficou na classe ORA. Ambas são destacadas por possuírem um número pequeno de árvores. Em especial, a classe ORA contém o menor número (=25) de toda base de dados utilizadas no treinamento, enquanto a classe OSV contém apenas 4 a mais.

5.2.4 Transferência de Aprendizagem para Bases Menores

Nesta etapa, utilizamos os modelos `densenet201` e `mobilenet_v2` obtidos na etapa anterior como base para uma transferência de aprendizagem nos conjuntos de dados `Trunk12` e `Bark-101`. Como estas bases possuem poucos dados, restringimos os experimentos de forma a treinar apenas a camada classificadora (densamente conectada).

Não exploramos técnicas como aumentação de dados e regularização L2. Entretanto, exploramos o melhor redimensionamento para essas bases, que traz a melhor acurácia com a `mobilenet_v2`, no conjunto de gráficos da Figura 9, resultado a ser replicado com a

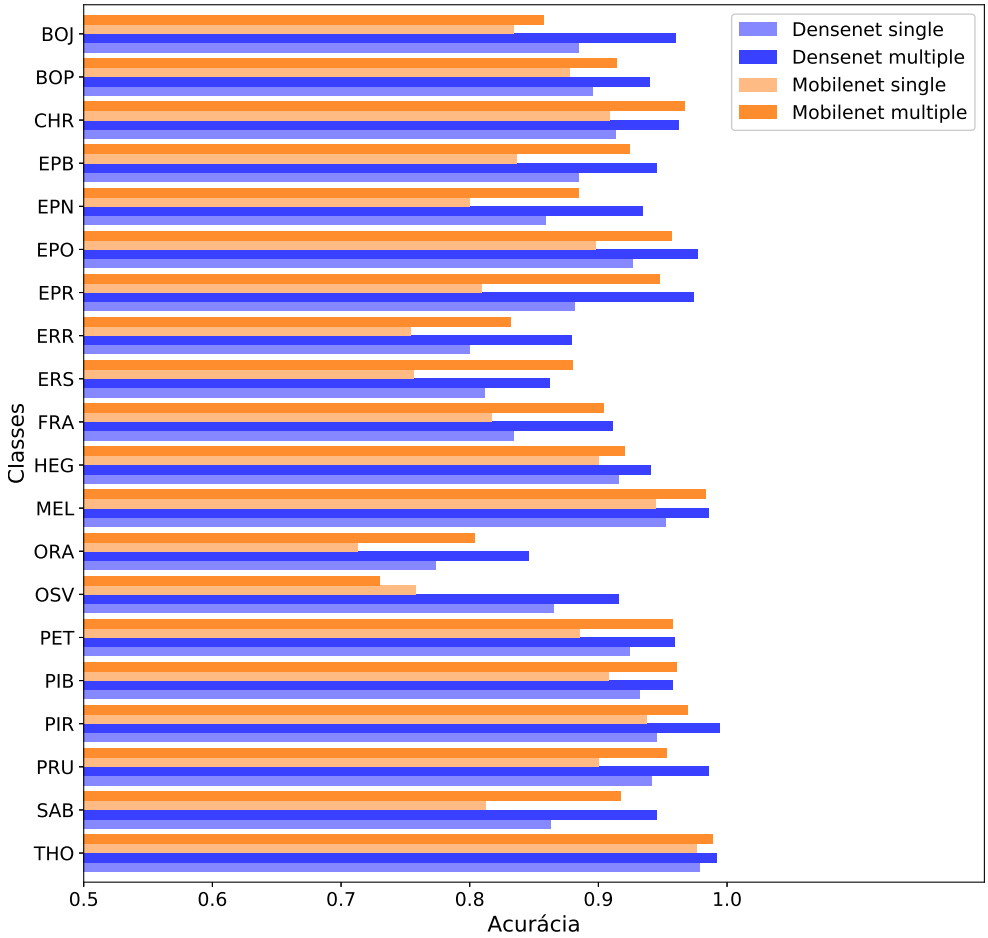


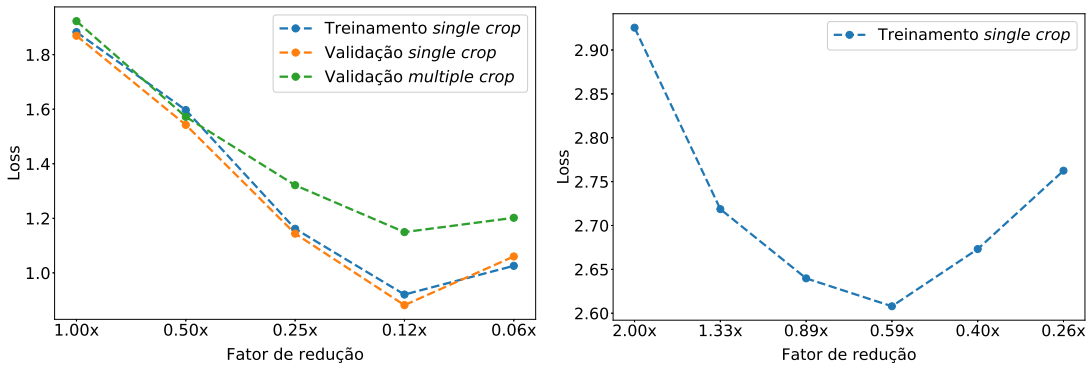
Figura 8: Acurácia final por classe dos modelos `mobilenet_v2` e `densenet201`.

`densenet201`.

Analisando a Figura 9, decidimos utilizar 0.12 como fator de redução para `Trunk12` e 0.59 para `Bark-101`, valores que apresentaram perda baixa. A Tabela 14 mostra o resultado com as arquiteturas `densenet201` e `mobilenet_v2` re-treinadas nas últimas camadas para a base de dados `Trunk12` e `Bark-101`.

Conforme pode ser observado na Tabela 14, os experimentos apresentaram bons resultados na base de dados `Trunk12`, condizente com o esperado, uma vez que essa base de dados, apesar de pequena em número de imagens para se realizar um treinamento do zero, é pouco desafiadora por conter imagens padronizadas na forma em que foram capturadas, sem variação em iluminação, escala e pouco ruído (Tabela 1). Por outro lado, para a base de dados `Bark-101`, temos um *overfitting* notável e esperado, dado o alto número de classes e ser mais desafiadora nos aspectos em que a `Trunk12` suaviza.

É notável também na Tabela 14 a pouca diferença nos valores de avaliação *single* e *multiple*. Isso pode ser explicado devido às dimensões das imagens utilizadas serem menores,



(a) Experimento com redimensionamento na `Trunk12`

(b) Experimento com redimensionamento na `Bark-101`

Figura 9: Gráficos da função de perda obtidos para alguns valores de redimensionamento das bases de dados `Trunk12` (Figura 9(a)) e `Bark-101` (Figura 9(b)), treinando apenas a camada classificadora do modelo `mobilenet_v2` pré-treinado para a `BarkNet`. Como algumas classes da `Bark-101` contêm pouquíssimas imagens, não foi feita uma divisão do conjunto em treinamento e validação.

Tabela 14: Resultado da transferência de aprendizado dos modelos `densenet201` e `mobilenet_v2` treinados para `BarkNet`, re-treinadas nas últimas camadas para a base de dados `Trunk12` e `Bark-101`. Todos modelos foram treinados utilizando 40 épocas, taxa de aprendizagem inicial 0.01, com decaimento de 5 nas épocas 15, 25 e 32, espelhamento horizontal como aumentação de dados e sem regularização L2. Para `mobilenet_v2` foi re-treinado as camadas `features.17`, `features.18` e *classifier*, e para `densenet201` apenas `classifier`. Na `Trunk12`, o protocolo foi validação cruzada com 5 partições, enquanto para `Bark-101` o protoco foi *hold-out* utilizando o protocolo próprio de treinamento e teste desta base de dados.

		Acurácia (%)			**Perda**	
	Arquitetura	**Treinamento** *single*	**Teste** *single*	**Teste** *multiple*	**Teste** *single*	**Teste** *multiple*
`Trunk12`	`densenet201`	99.53	96.43	95.41	0.1169	0.2323
	`mobilenet_v2`	98.82	95.41	96.17	0.1356	0.3206
`Bark-101`	`densenet201`	94.22	51.04	49.57	2.5202	2.5115
	`mobilenet_v2`	86.14	44.40	41.77	3.0441	3.0717

inviabilizando uma grande quantidade de *crops* no caso *multiple*.

6 Conclusões e Trabalhos Futuros

Neste trabalho, demonstramos a capacidade de se utilizar redes convolucionais profundas na identificação de espécies de árvores a partir de imagens de tronco. Experimentos foram

possíveis graças à base de dados `BarkNet`.

Procuramos explorar diversas arquiteturas, parâmetros de regularização, aumentação de dados e níveis de congelamento com pesos da `ImageNet`. O melhor resultado obtido foi 94.05% por imagem com protocolo de validação cruzada com 5 partições utilizando a `densenet201`. Entre as arquiteturas móveis, destacou-se também a `mobilenet_v2` sob o mesmo protocolo com acurácia de 91.47%.

A partir desses dois modelos para a *BarkNet*, experimentamos realizar uma transferência de aprendizagem para as bases de dados menores `Trunk12` e `Bark-101`. Entretanto, sem explorar exaustivamente hiperparâmetros e técnicas para melhorar a eficácia, conseguimos obter até 96.43% de acurácia no protocolo validação cruzada com 5 partições na base `Trunk12` e 51.04% utilizando o próprio conjunto de teste da base `Bark-101`.

Como trabalho futuro, uma proposta é explorar, de forma mais exaustiva, técnicas para melhorar a eficácia do processo de transferência de aprendizagem, principalmente para a base `Bark-101`, que é mais desafiadora.

Referências

[1] M. Carpentier, P. Giguère, and J. Gaudreault. Tree Species Identification from Bark Images Using Convolutional Neural Networks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1075–1081, Mar. 2018.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *International Conference on Computer Vision and Pattern Recognition*, 2009.

[3] S. Fiel and R. Sablatnig. Automated Identification of Tree Species from Images of the Bark, Leaves or Needles. Technical Report CVL-TR-3, Computer Vision Lab, Institute of Computer Aided Automation, Vienna University of Technology, Nov. 2010.

[4] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[5] A. Gron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 1st edition, 2017.

[6] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[7] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic Differentiation in PyTorch. In *31st Conference on Neural Information Processing System*, 2017.

[8] R. Ratajczak, S. Bertrand, C. Crispim-Junior, and L. Tougne. Efficient Bark Recognition in the Wild. In *International Conference on Computer Vision Theory and Applications*, pages 1–9, Feb. 2019.

[9] J. Schmidhuber. Deep Learning in Neural Networks: An Overview. *Neural Networks*, 61:85–117, 2015.

[10] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556*, 2014.

[11] N. Smolyanskiy, A. Kamenev, J. Smith, and S. Birchfield. Toward Low-Flying Autonomous MAV Trail Navigation using Deep Neural Networks for Environmental Awareness. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4241–4247, May 2017.

[12] M. Švab. *Computer-Vision-based Tree Trunk Recognition.* PhD thesis, Fakulteta za racunalništvo in informatiko, Univerza v Ljubljani, 2014.