UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Aplicação de deep learning em dispositivos Android

Tiago de Miranda Leite

Aplicação de deep learning em dispositivos Android

Tiago de Miranda Leite

Orientador: João do Espírito Santo Batista Neto

Monografia de conclusão de curso apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP - para obtenção do título de Bacharel em Ciências de Computação.

Área de Concentração: Inteligência Artificial

USP – São Carlos
Novembro de 2018

"O que sabemos é uma gota, o que ignoramos é um oceano."
– Sir Isaac Newton

Dedicatória

Dedico este trabalho aos meus pais, Maria José Ferreira da Luz Leite e Adelio Vaz de Miranda Leite, que sempre acreditaram no meu potencial e lutaram para dar uma boa educação a mim e a meus irmãos, não medindo esforços para que chegássemos até a Universidade. Aos meus irmãos João Paulo e Elis, por todo o afeto e companheirismo. Também aos vários amigos que fiz ao longo da vida, novos e de longa data, com quem sempre me divirto e aprendo algo.

Agradecimentos

Agradeço a todos os professores que tive na vida, pois tenho a certeza que cada um deles colaborou um pouco para que eu chegasse até aqui.

Minha gratidão também a todos que nunca deixaram de acreditar no meu potencial, mesmo nos momentos mais difíceis desta jornada.

Por fim, agradeço à Universidade de São Paulo e ao Instituto de Ciências Matemáticas e de Computação, sem os quais não seria possível a realização deste trabalho.

Resumo

Modelos *deep learning* têm se mostrado eficazes na resolução de diversos problemas. Especificamente, em classificação de imagens, as redes neurais convolucionais são o modelo mais bem sucedido. Embora seu treinamento normalmente exija grande quantidade de processamento e memória, há modelos que funcionam de maneira eficiente em ambientes com pouca disponibilidade desses recursos, como os *smartphones*. O objetivo deste trabalho é o treinamento e posterior utilização de um desses modelos, a rede convolucional MobileNet, em um aplicativo para sistema Android com a função de reconhecimento de espécies de flores em fotos. Para tanto, algumas configurações existentes da rede foram treinadas e avaliadas, considerando aspectos como acurácia e tamanho, entre outros, a fim de se obter a opção mais adequada para o problema. Para o treinamento, utilizou-se a linguagem Python e a biblioteca TensorFlow, enquanto na implementação do aplicativo foi empregada a linguagem Java. O modelo escolhido obteve acurácia de 0.955 no conjunto de teste enquanto possuía tamanho de apenas 10,2 MB. O aplicativo resultante permite a classificação da imagem fornecida pelo usuário dentre as 16 classes de espécies utilizadas no treinamento, com todo o processamento ocorrendo diretamente no dispositivo.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Gráficos

CAPÍTULO 1: INTRODUÇÃO

1.1. Contextualização e Motivação

Métodos de aprendizado de máquina baseados em *deep learning* têm se mostrado eficazes em diversas tarefas como análise de sinais de áudio, previsão de séries temporais, processamento de linguagem natural e, especialmente, em classificação de imagens, que é o foco deste trabalho.

Redes neurais convolucionais têm sido o modelo de *deep learning* mais utilizado nesse tipo de problema. Embora resultados importantes tenham sido obtidos com a rede LeNet-5 (LECUN et al., 1998), foi somente a partir dos últimos anos que tais modelos ganharam notabilidade, principalmente com a apresentação da rede AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) e seu notável desempenho na competição ILSVRC (*ImageNet Large-Scale Visual Recognition Challenge*). Além do mais, avanços em tecnologia de fabricação de hardware de processamento paralelo, como as unidades de processamento gráfico (GPU), tornaram viáveis os treinamentos de modelos cada vez maiores e sofisticados.

Simultaneamente, observa-se, ao longo da última década, uma popularização do uso de dispositivos móveis, como *smartphones* e *tablets*, bem como seus diversos aplicativos. É possível encontrar nas lojas virtuais Google Play e App Store diversos aplicativos que utilizam inteligência artificial em imagens para detecção de rostos e aplicação de filtros artísticos, por exemplo. Há também alguns que identificam espécies vegetais através de fotos, como o PictureThis e o PlantNet, porém tais aplicações necessitam de conexão à internet, visto que o processo de identificação ocorre em servidor remoto.

Há, no entanto, modelos de redes otimizados para rodarem no próprio dispositivo móvel, como a rede MobileNet (HOWARD et al., 2017). Assim, tais modelos podem ser utilizados para realizar todo o processamento de identificação da imagem localmente. Essa característica torna-se interessante no contexto da aplicação, uma vez que o usuário não encontraria problemas ao utilizar o aplicativo numa região sem disponibilidade de internet, como um parque, um bosque ou qualquer região com problemas de sinais de internet móvel.

1.2. Objetivos

Este trabalho tem como objetivo a criação de um aplicativo para dispositivos com sistema Android que realize a classificação de espécies de plantas através de imagens de suas flores, fornecidas pelo usuário, através de câmera ou de fotos previamente obtidas. Além disso, o aplicativo deverá mostrar um histórico de todas as imagens já identificadas, dispostas em forma de linha do tempo, em sua tela principal, além de fornecer informações adicionais sobre a espécie encontrada e opções de compartilhamento da imagem nas redes sociais.

Para tanto, será empregada a rede neural convolucional MobileNet, a qual, rodando diretamente no aplicativo, executará localmente todo o processo de classificação da imagem. Para que a rede seja capaz de realizá-lo corretamente, será necessário, primeiramente, executar seu treinamento utilizando-se um conjunto de dados com diversas imagens rotuladas de flores.

Diversas variações da arquitetura da rede MobileNet serão treinadas e analisadas, a fim de se obter o modelo mais adequado para o problema em questão, a partir da análise das medidas de performance de classificação. Uma vez escolhido, tal modelo será transferido ao aplicativo.

1.3. Organização da Monografia

O Capítulo 2 apresenta o conhecimento teórico necessário ao desenvolvimento do projeto, bem como diversos trabalhos da literatura que serviram de apoio à sua realização. No Capítulo 3 são descritas todas as atividades realizadas, considerando-se aspectos como metodologia, tecnologias usadas, linguagens de programação e tipos de dados; ao final são exibidos os resultados obtidos, bem como são discutidas as limitações e sugestões de trabalhos futuros. Finalmente, no Capítulo 4 é feita a conclusão do projeto, destacando suas contribuições, além de uma análise a respeito do curso de graduação no qual se dá o contexto deste trabalho.

CAPÍTULO 2: REVISÃO BIBLIOGRÁFICA

2.1. Considerações Iniciais

Neste capítulo são apresentados os principais conceitos necessários ao entendimento e desenvolvimento do trabalho. Inicialmente é relatado como se deu o avanço das arquiteturas de redes neurais através dos anos, até os dias atuais; destaque especial é dado às redes convolucionais e seu estabelecimento como modelo *deep learning*. Por fim, apresenta-se a estratégia de transferência de conhecimento e como tal abordagem pode ajudar no treinamento dos modelos.

2.2. Evolução das Redes Neurais: do Primeiro Modelo ao *Deep Learning*

A proposta do primeiro modelo matemático de um neurônio biológico remonta à década de 1940. McCulloch e Pitts (1943) propuseram uma maneira modelar o processo de aprendizado humano através da criação de modelos que funcionam como portas lógicas, sendo capazes então, em combinação, de representar funções *booleanas*.

Rosenblatt (1957) apresentou um classificador binário denominado Perceptron, um modelo de neurônio com parâmetros (pesos) variáveis, podendo ser treinado através do uso de algoritmo de aprendizado. Tal modelo mostrou-se capaz de resolver problemas de classificação no qual os exemplos positivos e negativos pudessem ser separados por um hiperplano. Entretanto, notou-se que o Perceptron era incapaz de representar o funcionamento de uma simples porta lógica XOR, o que gerou desapontamentos e tornaria as redes neurais praticamente esquecidas pelos próximos anos.

Embora já houvesse a ideia de combinar neurônios em múltiplas camadas, não se sabia como realizar seu treinamento, já que não havia uma maneira clara de como alterar os parâmetros dos neurônios presentes nas camadas intermediárias. Para solucionar tal problema, Rumelhart, Hinton e Williams (1986) apresentaram o algoritmo de aprendizado *backpropagation*, conferindo às redes neurais a capacidade de resolver problemas não lineares, reavivando interesses na área.

Uma das primeiras redes convolucionais foi introduzida por LeCun et al. (1998), denominada LeNet-5, a qual obteve resultados notáveis em classificação de imagens de dígitos manuscritos, utilizando o conjunto de dados MNIST. A partir de meados da década de 2000, o termo *deep learning* começou a ganhar popularidade, principalmente com as criação das redes profundas de treinamento não supervisionado (HINTON; OSINDERO; TEH, 2006) e as redes *autoencoder* (HINTON; SALAKHUTDINOV, 2006).

A partir de então, redes cada vez maiores e mais profundas têm sido implementadas. Em 2012, a rede AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) venceu o concurso ILSVRC por uma grande margem. Tal fato destacou o poder das redes convolucionais como modelo *deep learning*, servindo de base para diversas outras arquiteturas de redes que surgiriam.

2.3 Redes Convolucionais

Pode-se dizer que as redes neurais convolucionais correspondem ao estado da arte para problemas de classificação de imagens, sendo o modelo *deep learning* mais utilizado nesse âmbito. Tendo sido inspiradas no funcionamento das células do córtex visual dos animais, elas tentam tirar vantagem do arranjo espacial dos dados, através de operações de convolução.

Assim, numa camada de convolução, os neurônios são reunidos em conjuntos denominados mapas, que podem ser interpretados como filtros que serão aplicados à imagem. Tais filtros acabam por varrer toda a imagem, produzindo como saída um mapa de atributos, que corresponde à imagem de entrada apresentando algum aspecto realçado, como bordas, texturas ou linhas horizontais, por exemplo. Para que o mesmo filtro possa ser aplicado a diferentes regiões da imagem, é necessário que os neurônios de cada mapa apresentem os mesmos pesos, o que contribui para diminuir o número de parâmetros da rede, permitindo que seja treinada mais rapidamente.

O treinamento da rede corresponde a encontrar valores para os filtros que melhor servirão para agrupar as imagens em suas respectivas classes, utilizando para isso uma variação do algoritmo *backpropagation* (PATTANAYAK, 2017). Além disso, normalmente após cada camada de convolução, aplica-se uma camada de *pooling* responsável por diminuir as dimensões dos dados de entrada.

Camadas de convolução e *pooling* podem ser arranjadas de maneira sequencial, com a saída de uma correspondendo à entrada da próxima, permitindo a obtenção de mapas de características que destacam atributos cada vez mais complexos e abstratos, conforme sua profundidade na rede. Essas sucessivas representações parciais dos dados de entrada, organizadas dessa maneira hierárquica, conferem o aspecto *deep learning* a esse tipo de modelo.

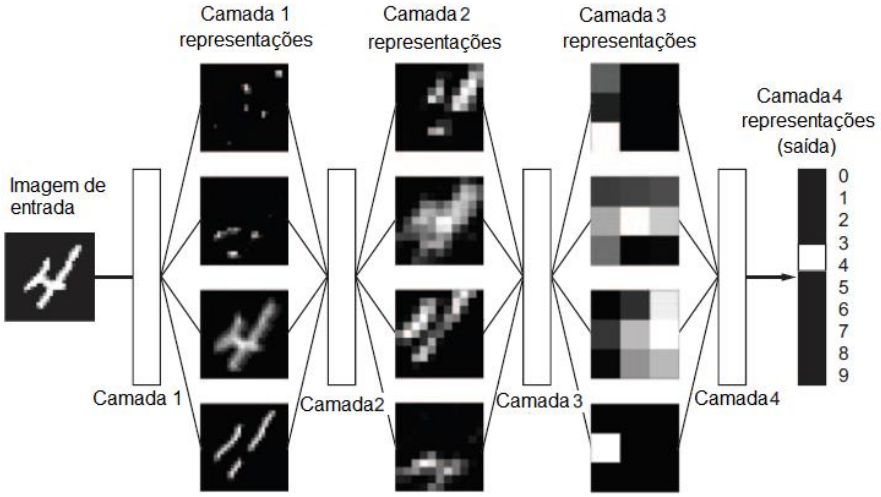

Figura 1: Diferentes atributos da imagem extraídos por cada camada de convolução. Fonte: Adaptada de Chollet (2018)

2.3.1 MobileNet

Após o sucesso da AlexNet, diversas arquiteturas de redes foram propostas nos últimos anos; no entanto, tais redes demandam enorme poder computacional tanto para seu treinamento quanto no processo de predição. Com o crescente uso de dispositivos móveis, como *smartphones* e *tablets*, fez-se necessária a criação de modelos que funcionassem dentro das limitações de processamento e memória presentes em tais dispositivos, permitindo a criação de aplicativos inteligentes. Uma das mais bem sucedidas redes nesse contexto é a MobileNet (HOWARD et al., 2017), que apresenta algumas variantes relacionadas aos tamanhos dos mapas de atributos em cada camada de convolução, permitindo ao programador escolher qual versão mais adequada à sua aplicação.

2.4 Transferência de Conhecimento

Modelos *deep learning* normalmente possuem milhares ou milhões de parâmetros a serem treinados, o que demanda grande quantidade de tempo e processamento caso o treinamento seja executado a partir do zero, ou seja, com tais parâmetros iniciados de maneira aleatória. No âmbito de classificação de imagens e redes convolucionais, uma determinada rede já treinada com um extenso conjunto de imagens poder ser reutilizada, a partir do ponto onde parou, em um novo treinamento com outro conjunto de imagens.

Uma vez que camadas de convolução iniciais extraem características de mais baixo nível, como formas, contornos e texturas, opta-se por não mais alterar seus pesos, de modo que somente os pesos das camadas finais da rede são modificados, a fim de torná-los mais refinados ao novo conjunto de dados em questão. Seguindo essa abordagem, podem ser obtidos resultados melhores ao utilizar tal técnica, quando comparada ao treinamento realizado a partir do zero (OQUAB et al., 2014).

2.5. Considerações Finais

Neste capítulo foi apresentada uma visão geral a respeito dos modelos de redes neurais existentes ao longo do tempo, até as atuais redes convolucionais, inseridas no contexto *deep learning*, além de destacar a utilidade do uso de transferência de conhecimento no treinamento de tais modelos. Por fim, destacou-se a rede MobileNet, a qual foi utilizada como base no desenvolvimento deste trabalho.

CAPÍTULO 3: DESENVOLVIMENTO DO TRABALHO

3.1. Considerações Iniciais

Este capítulo apresenta todo o desenvolvimento do trabalho, iniciando por uma descrição do problema abordado, para então detalhar as atividades realizadas na busca de sua solução e, finalmente, os resultados alcançados. Inclui-se o detalhamento de linguagens de programação, bibliotecas, técnicas e conjunto de dados que foram utilizadas.

3.2. Descrição do Problema

O objetivo deste trabalho é a criação de um aplicativo inteligente para dispositivos Android capaz de identificar algumas espécies de flores através de imagens. Conforme já mencionado, para que isso fosse possível, foi utilizada a rede neural MobileNet e algumas de suas variantes, previamente treinadas com o extenso banco de imagens ImageNet (DENG et al., 2009) para então, por um processo de transferência de conhecimento, serem novamente treinadas com um conjunto de imagens específico contendo as espécies de flores selecionadas. Através da análise do desempenho de cada variante da rede no conjunto de testes, foi realizada a seleção do modelo final, o qual foi enviado para rodar no aplicativo Android.

A linguagem de programação utilizada na fase de treinamento foi Python (versão 3.5), bem como sua biblioteca TensorFlow (versão 1.10.0), ambas muito utilizadas atualmente em projetos de aprendizado de máquina. Quanto ao aplicativo, a linguagem utilizada foi Java, que ainda é a mais utilizada em desenvolvimento Android.

3.3. Descrição das Atividades Realizadas

Nesta seção será descrito cada procedimento executado durante o desenvolvimento do projeto, abrangendo as estratégias envolvidas desde a obtenção inicial do conjunto de dados, escolha e treinamento dos modelos, até a elaboração final do aplicativo.

3.3.1. Obtenção da conjunto de imagens

Inicialmente foi feito um levantamento sobre as espécies mais comuns de flores encontradas nas proximidades do câmpus, tendo sido então consideradas 15 dessas espécies mais frequentemente encontradas. Foram elas, em sua denominação popular: Margarida, Primavera, Girassol, Hibisco, Ixora, Lantana, Narciso, Oleandro, Bela Emília, Azaleia, Rosa, Estrelítzia, Dente-de-leão, Amor-Perfeito e Olho-Preto. Para cada uma dessas classes, foi realizada uma busca de exemplares no site Google Imagens, realizando-se o *download* das melhores amostras, isto é, aquelas em que a flor em questão ocupava a maior parte de imagem, com tamanho de no mínimo 200 pixels, sendo essa inspeção realizada de maneira visual.

Há um conjunto de dados da Universidade de Oxford, consistindo de 8189 imagens de flores distribuídas ao longo de 102 classes (NILSBACK; ZISSERMAN, 2008). Azaleia, Hibisco, Dente-de-leão, Girassol, Rosa e Narciso estão entre as espécies presentes, tendo sido então adicionadas ao conjunto de dados. Por fim, uma última classe foi adicionada, com imagens que não continham flores, para que a rede pudesse identificar fotos que não continham nenhuma das categorias anteriores.

Assim, havia ao todo 16 classes, cada uma contendo de 250 a 300 imagens, totalizando 4713 arquivos no formato png ou jpg.

3.3.2. Obtenção da rede

Diversos modelos *open source* da rede MobileNet, já treinados com o conjunto de imagens ImageNet, podem ser baixados do Google AI Blog (HOWARD; ZHU, 2017). As diversas versões dessa rede variam parâmetros como o tamanho da imagem de entrada e o número de mapas de atributos de cada camada de convolução da rede. Para o tamanho da imagem, as opções são, em pixels, 128x128, 160x160, 192x192 e 224x224; no entanto, decidiu-se por somente utilizar a versão com maior tamanho de imagem, para que posteriormente tal imagem pudesse ser mostrada na tela do aplicativo com a menor perda possível causada pelo redimensionamento. Quanto ao número de mapas, cada versão apresenta uma fração do número de mapas de atributos de cada camada convolução da configuração original, com valores podendo corresponder a 25%, 50%, 75% e 100%.

O site Google Developers Codelabs (https://codelabs.developers.google.com) apresenta diversos tutoriais a respeito da criação de interessantes projetos relacionados às suas diversas de suas ferramentas e produtos, como Android, Google Cloud, Firebase e Google Analytics. O ponto de partida deste projeto foi o tutorial “TensorFlow for Poets”, o qual fornece um link para um repositório do Google no Github (https://github.com/googlecodelabs/tensorflow-for-poets-2), contendo os códigos inicias para *download* e treinamento da rede MobileNet. Foi então realizado um *fork* desse repositório, para alteração dos códigos de modo a se adaptarem às necessidades do projeto.

O único arquivo efetivamente utilizado foi o script *retrain.py*, responsável pelo *download* e treinamento da rede. A primeira alteração foi a criação de uma nova camada no fim da rede, de tamanho 16 (número de classes do novo conjunto de dados sendo utilizado), totalmente conectada à última camada da rede original, com função de ativação *softmax*. Na figura e seguir, há uma esquematização do modelo, em que *conv* refere-se a uma camada de convolução, *fc* a uma camada totalmente conectada e os números entre parênteses indicam tamanho de camada (número de neurônios).

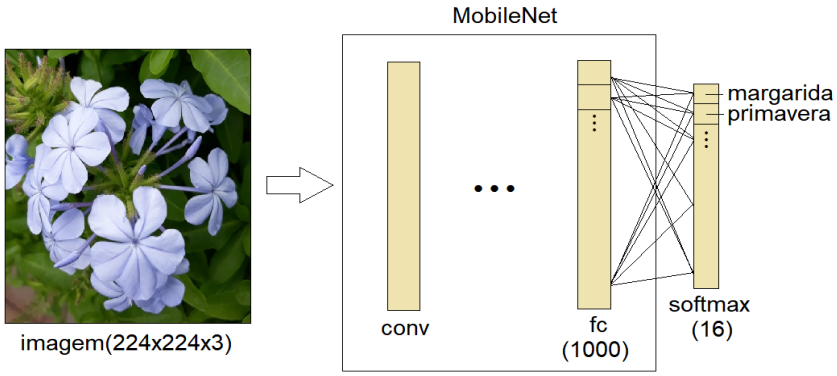

Figura 2: Esquema da arquitetura inicial de rede MobileNet utilizada. Fonte: elaborada pelo autor.

Para fins de comparação, foi criado mais um modelo de arquitetura da rede, ao se adicionar duas camadas totalmente conectadas, uma com 512 neurônios, com função de ativação *ReLU* e usando *dropout*, já que esta última tem sido uma abordagem muito utilizada para evitar sobreajuste, e finalmente a última camada com função de ativação *softmax*, com 16 neurônios; tal função de ativação é tipicamente utilizada apenas na camada final, pois

permite a interpretação dos valores que retorna como uma distribuição de probabilidades entre as classes sendo previstas.

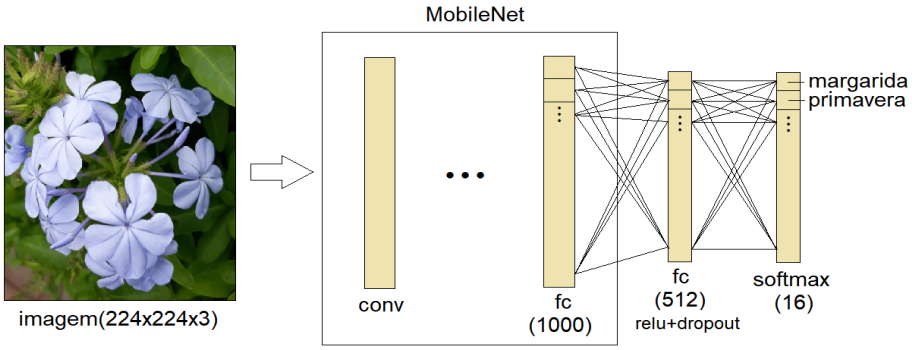

Figura 3: Esquema de outra arquitetura testada. Fonte: elaborada pelo autor.

Assim, havia ao todo 8 arquiteturas da rede para serem treinadas e avaliadas: aquela ilustrada na Figura 2, com suas 4 variações de tamanho dos mapas de atributos (25%, 50%, 75% e 100%), e a da Figura 3, também com essas quatro variações. Tais arquiteturas estão organizadas e resumidas na tabela a seguir:

Grupo	Nome	Fração dos mapas de atributos	Camadas adicionadas	*Dropout*
1	MobileNet_1_25	25%	softmax (16)	Não
1	MobileNet_1_50	50%	softmax (16)	Não
1	MobileNet_1_75	75%	softmax (16)	Não
1	MobileNet_1_100	100%	softmax (16)	Não
2	MobileNet_2_25	25%	fc(512) e softmax (16)	Sim
2	MobileNet_2_50	50%	fc(512) e softmax (16)	Sim
2	MobileNet_2_75	75%	fc(512) e softmax (16)	Sim
2	MobileNet_2_100	100%	fc(512) e softmax (16)	Sim

Tabela 1: Modelos de rede testados e suas características.

3.3.3. Treinamento

Cada uma das 8 arquiteturas foi treinada com 70% do conjunto de dados coletado, havendo 10% para validação e restando os outros 20% para teste; ressalta-se que somente as camadas adicionadas foram treinadas, mantando-se congeladas as camadas internas da rede, para tirar proveito do uso de transferência de conhecimento. O otimizador utilizado em todas elas foi o AdamOptimizer, muito utilizando atualmente e já fornecido pela biblioteca TensorFlow, com função de erro (custo, ou *loss*) sendo a entropia cruzada. Cada iteração de treinamento utilizou um lote (*batch*) de 100 imagens aleatórias. Para as arquiteturas com

duas camadas adicionais, utilizou-se taxa de *dropout* de 50% na penúltima camada, conforme já mencionado, o que significa que, em cada iteração, cada neurônio daquela camada tem uma probabilidade de 50% de não ter seus pesos alterados.

Os valores do erro da cada rede e da acurácia no conjunto de validação, durante o treinamento, foram acompanhados através da interface gráfica *tensorboard*, recurso da biblioteca TensorFlow que permite visualização de gráficos em tempo real, apresentados na tela do navegador e que serão apresentados na seção 3.4.

3.3.4. Implementação do Aplicativo

Após o treinamento, salvou-se a rede treinada na forma de um arquivo com a extensão .pb (*protobuffer*), que é o formato de arquivo utilizado pelo TensorFlow para permitir a posterior execução da rede neural em outras plataformas. O aplicativo foi implementado utilizando-se a ferramenta Android Studio, que é o ambiente oficial para desenvolvimento para aplicações para dispositivos Android, utilizando-se a linguagem Java.

Para poder fazer o carregamento e uso da rede neural, utilizou-se a versão do TensorFlow para Android, bastando, para isso, adicionar as dependências ao Gradle, que é o sistema gerenciador de dependências do Android Studio. Com isso, seria possível enviar à rede, como entrada, a imagem escolhida pelo usuário, seja pela câmera ou pela galeria de fotos, para que fosse feita a classificação.

Para que o aplicativo mantivesse o histórico das imagens, utilizou-se Realm, um banco de dados orientado a objetos e feito exclusivamente para ambiente *mobile*, tornando-o uma boa alternativa ao tradicional banco de dados relacional SQLite, nativo do Android.

3.4. Resultados Obtidos

3.4.1 Treinamentos e arquitetura escolhida

Inicialmente, foram realizados os treinamentos das quatro redes do Grupo 1, durante 5000 iterações. Os valores das acurácias de cada modelo, ao longo das iterações de treinamento, estão no gráfico a seguir:

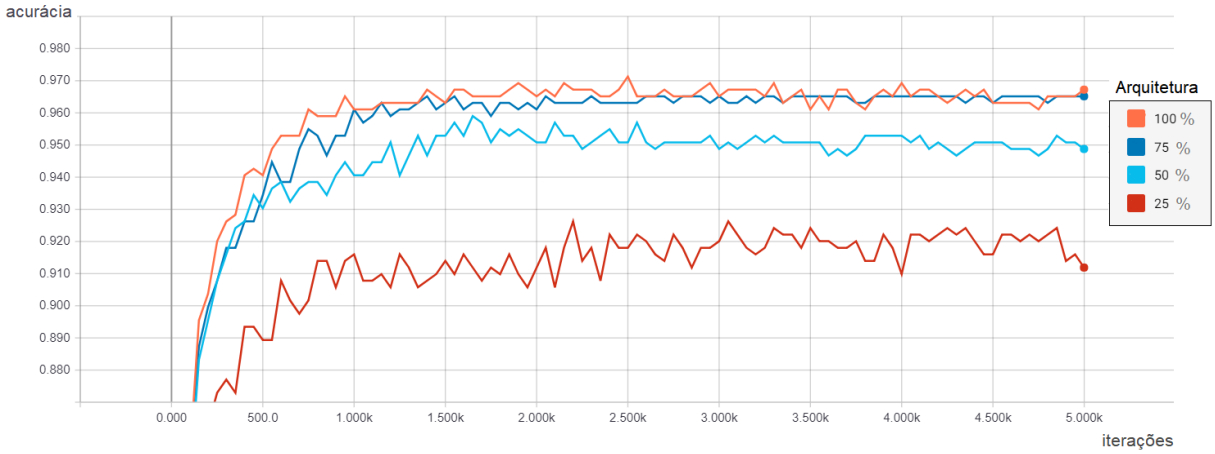

Gráfico 1: Valores de acurácia das redes do grupo 1 ao longo do treinamento.

Observa-se que as arquiteturas MobileNet_1_75 e MobileNet_1_100 tiveram os maiores valores de acurácia e ficaram empatadas, em torno de 0,965. Posteriormente, foi realizado novo treinamento, utilizando-se as arquiteturas do Grupo 2, sendo mantidas as demais condições do primeiro treinamento. Resultados no gráfico a seguir:

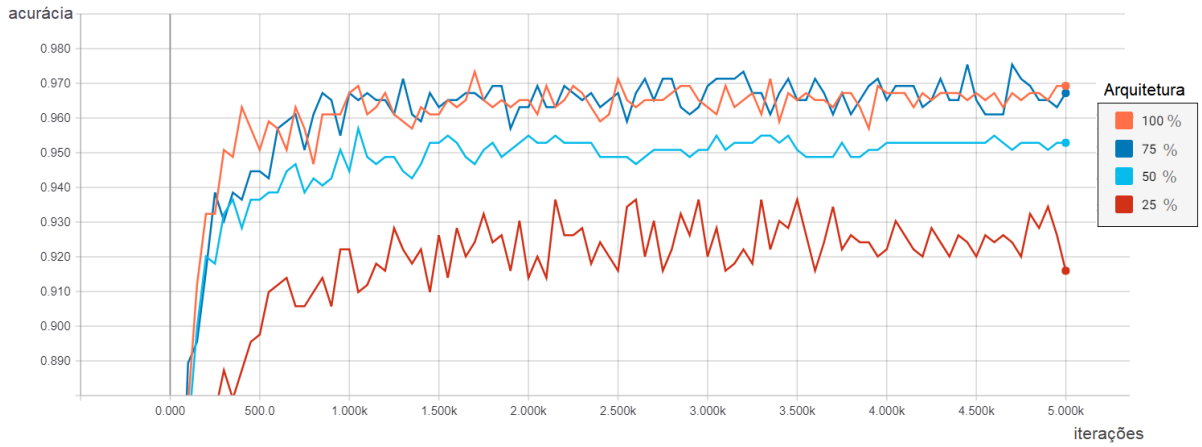

Gráfico 2: Valores de acurácia das redes do grupo 2 ao longo do treinamento.

Novamente, constata-se empate entre a mesmas duas arquiteturas do primeiro treinamento, embora desta vez haja mais oscilações devido ao *dropout*.

Treinamentos com maior número de iterações foram realizados, mas os gráficos resultantes foram essencialmente os mesmos que os apresentados. Na verdade, nota-se uma certa estabilidade dos valores de acurácia a partir da iteração 3000.

Além da acurácia, outras métricas costumam ser utilizadas para avaliação de algoritmos de classificação, como precisão, *recall* (ou cobertura) e *F-score* (SOKOLOVA; LAPALME, 2009). Assim, para uma dada classe C_i de um total de *k* classes, sendo tp_i o número de verdadeiros positivos, fp_i os falsos positivos, tn_i os verdadeiros negativos e fn_i o número de falsos negativos, define-se:

$$acurácia = \frac{\sum_{i=1}^{k} \frac{tp_i + tn_i}{tp_i + fp_i + tn_i + fn_i}}{k}$$

$$precisão = \frac{\sum_{i=1}^{k} \frac{tp_i}{tp_i + fp_i}}{k}$$

$$recall = \frac{\sum_{i=1}^{k} \frac{tp_i}{tp_i + fn_i}}{k}$$

$$F - score = \frac{2 * precisão * recall}{precisão + recall}$$

Tais valores foram calculados para as duas melhores arquiteturas de cada grupo, usando-se o conjunto de teste, sendo apresentados na tabela a seguir. Neste caso, devido às limitações de espaço nos dispositivos móveis, também foi considerado o tamanho final do arquivo .pb no qual a rede foi salva.

Nome	**Acurácia**	**Precisão**	***Recall***	***F-score***	**Tamanho**
Mobilenet_1_100	0,954	0,955	0,953	0,954	17,2 MB
Mobilenet_2_100	0,961	0,961	0,959	0,960	19,2 MB
Mobilenet_1_75	0,955	0,956	0,956	0,956	10,6 MB
Mobilenet_2_75	0,955	0,957	0,956	0,956	12,6 MB

Tabela 2: Valores de acurácia, precisão, recall, F-score e tamanho das arquiteturas escolhidas.

Nota-se que as redes do grupo 1 tiveram suas métricas muito próximas às suas respectivas do grupo 2, mas com tamanho final menor. A arquitetura Mobilenet_1_75, por ser a de menor tamanho (apenas 10,6 MB) entre as melhores, e ter valores de avaliação muito próximos às demais, mostrou-se a melhor opção, sendo a escolhida para compor o aplicativo. A matriz de confusão da arquitetura escolhida encontra-se no Apêndice A.

3.4.2 Aplicativo

O aplicativo foi sendo implementado concomitantemente à realização dos treinamentos, para que já pudesse ir sendo testado ao longo do tempo, evitando-se deixar muitas tarefas de implementação para o final. Conforme já mencionado, a versão do TensorFlow para Android fornece uma biblioteca de alto nível em Java para carregar e rodar um modelo através do arquivo .pb no qual a rede fora salva. Fornecendo à rede a imagem obtida pelo usuário, na forma de um *array* de inteiros que representam os pixels, retorna-se um *array* de 16 posições, correspondente à camada de saída da rede que fora criada, indicando as probabilidades de cada classe.

Algumas capturas de tela do aplicativo são mostradas na Figura 4 a seguir. À esquerda, a tela principal, na qual as imagens de flores já utilizadas e identificadas são distribuídas na forma de uma linha do tempo vertical; novas imagens são carregadas à medida que o usuário faz a rolagem da tela para baixo. Para cada foto há um botão para seu compartilhamento em redes sociais e em outros aplicativos. Na parte inferior da tela existe um botão flutuante para adição de novas fotos, vindas da câmera ou da galeria de fotos do dispositivo. No centro, a tela que mostra as espécies identificadas, com suas probabilidades, para o usuário escolher; apenas classificações com probabilidade maior ou igual a 10% são apresentadas. Por fim, à direita, uma janela fornecendo outros exemplos de imagens da mesma espécie que a encontrada, para o usuário realizar inspeção visual e confirmar a veracidade da classificação.

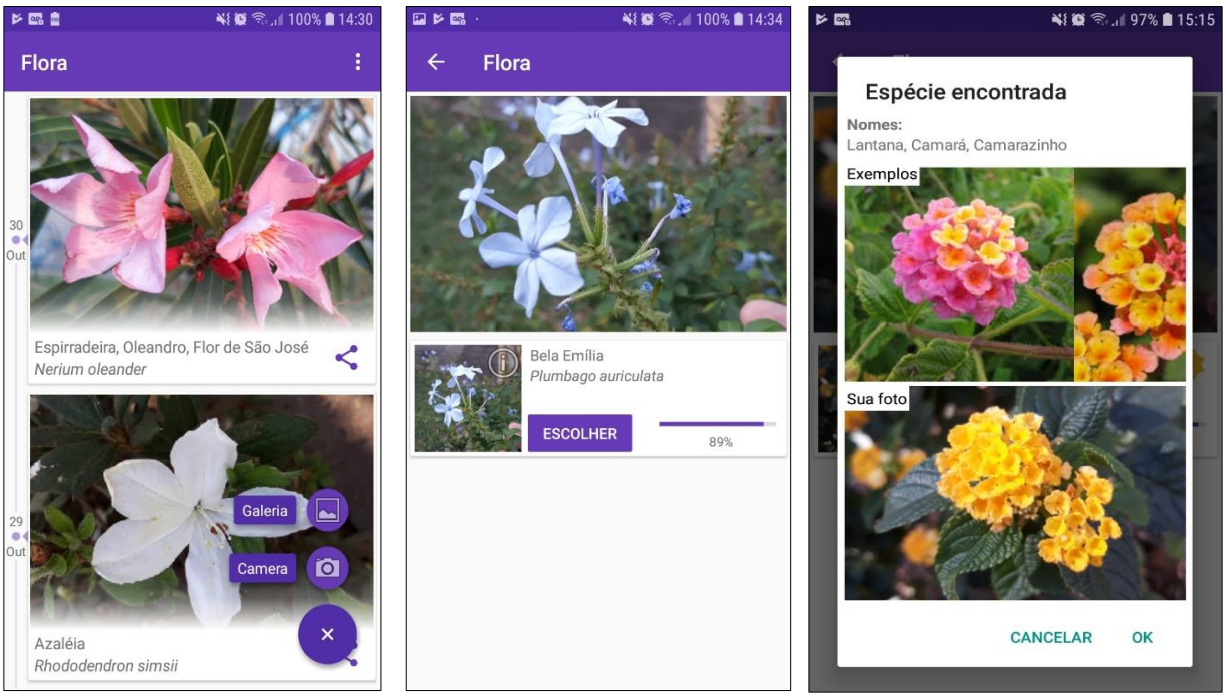

Figura 4: Capturas de tela do aplicativo. Fonte: elaborada pelo autor.

Quando o usuário clica em alguma imagem na tela principal, mostra-se uma outra tela, com informações biológicas mais detalhadas a respeito de espécie encontrada, como nome científico, família, gênero, regiões de ocorrência, entre outros. Tais dados foram obtidos a partir de um arquivo no formato csv, disponibilizado para *download* pelo Portal Brasileiro de Dados Abertos, contendo diversas informações sobre espécies vegetais do Brasil (LISTA DE ESPÉCIES DA FLORA DO BRASIL, 2015). O usuário pode, ainda, deletar a foto ao clicar no ícone de lixeira na parte inferior direita da imagem.

Figura 5: Captura de tela apresentando informações detalhadas sobre a espécie. Fonte: elaborada pelo autor.

3.5. Dificuldades, Limitações e Trabalhos Futuros

As principais dificuldades relacionam-se ao fato de não se dispor do *hardware* comumente utilizado para treinamento de redes neurais, que são as GPUs; todo o treinamento foi realizado no processador (CPU), levando horas até terminar. A principal consequência de tal fato é que diferentes valores de parâmetros como tipo de otimizador, taxas de *dropout*, funções de ativação, taxas de aprendizado e número de neurônios não puderam ser amplamente testados. Poder-se-ia, além disso, treinar algumas camadas finais de convolução, em vez de refinar somente a camada totalmente conectada, caso houvesse maior poder computacional disponível.

Outra dificuldade foi estabelecer o pleno funcionamento da rede em ambiente Android. Embora tenha sido utilizada biblioteca pronta, a mudança de ambiente provocou muitos erros durante a execução do aplicativo, devido principalmente às diferentes representações dos tipos de dados que as linguagens Java e Python realizam internamente.

Futuramente, pode-se ampliar o número de espécies identificadas, acrescentando-se mais classes ao conjunto de treinamento, além de oferecer ao usuário a possibilidade de contribuir com a aplicativo, ao enviar suas fotos ao banco de imagens de treinamento.

3.6. Considerações Finais

Neste capítulo foi exposto todo o desenvolvimento do trabalho, contemplando aspectos das ferramentas, dados e metodologias utilizadas. Fez-se uma análise do desempenho de cada modelo testado e, por fim, destacou-se a implementação e funcionamento do aplicativo resultante, bem como as limitações e sugestões de continuidade do trabalho.

CAPÍTULO 4: CONCLUSÃO

4.1. Contribuições

Este trabalho mostrou que é possível a reutilização, refinamento e manipulação de modelos de redes neurais convolucionais, pré-treinados, para classificação de imagens aplicadas a um contexto mais específico. Permitiu, também, a observação de que nem sempre o aumento do número de parâmetros de uma rede necessariamente acarreta melhoras no desempenho de classificação, e que fatores decorrentes desse aumento também devem ser considerados, como o tamanho final do modelo, principalmente num ambiente com disponibilidade limitada de espaço e processamento, tal como são os *smartphones*. Além disso, demonstrou a possibilidade de se utilizar o modelo em ambiente Android, através de um aplicativo que pode ser utilizado por qualquer pessoa, em qualquer lugar do mundo.

O alto grau de conhecimento interdisciplinar adquirido pelo autor foi uma das principais contribuições pessoais obtidas. Além da área de inteligência artificial, o projeto também contemplou o ramo de processamento de imagem e, para a criação do aplicativo, noções de interface humano-computador e *design* foram importantes. Outros tópicos como *multithreading*, padrões de projeto orientado a objetos e banco de dados também mostraram-se presentes durante a implementação do aplicativo. Até mesmo outras áreas do conhecimento, como botânica e morfologia do neurônio biológico puderam ser aprendidas ao longo do desenvolvimento deste trabalho.

4.2. Considerações sobre o Curso de Graduação

O curso de Bacharelado em Ciências de Computação da Universidade de São Paulo no campus de São Carlos forneceu ao autor sólidas bases de conhecimento para a criação deste trabalho. Enquanto disciplinas teóricas permitiram-lhe a compreensão dos aspectos mais abstratos e matemáticos da computação que alicerçaram todo o conhecimento que seria adquirido, a frequente existência de trabalhos práticos estimulou o espírito de desenvolvimento de algo mais aplicado, a partir da teoria.

Especificamente, a disciplina de Introdução à Ciência da Computação I, do início do curso, forneceu conhecimento utilizado até hoje, como estruturas condicionais e de repetição, além de lógica de programação. Igualmente importantes foram Introdução à Ciência da Computação II e Algoritmos e Estruturas de Dados, que além de refinarem o raciocínio lógico para programação, forneceram conceitos valiosos que todo bom desenvolvedor deve manter em mente, como complexidade de algoritmos, estruturas de dados, uso eficiente de memória computacional e otimização. Pela mesma razão, também foi de grande ajuda a disciplina de Algoritmos Avançados e Programação Matemática.

A disciplina de Programação Orientada a Objetos forneceu os conhecimentos iniciais para programação em linguagem Java, utilizada em desenvolvimento Android. Todo o interesse do autor na área de aprendizado de máquina exacerbou-se quando da realização da disciplina Introdução a Redes Neurais, cujos trabalhos práticos também permitiram evoluir seu conhecimento no uso da linguagem Python e da biblioteca TensorFlow. Banco de Dados e Programação Concorrente também marcaram presença, ainda que de maneira secundária, no desenvolvimento do projeto.

Uma sugestão para o curso refere-se à existência de alguma disciplina da área de humanidades, principalmente relacionada à comunicação e expressão, voltada a desenvolver as habilidades dos alunos na criação de diversos tipos de apresentações, sejam orais ou escritas, bem como elaboração de discursos coerentes e argumentação.

REFERÊNCIAS

CHOLLET, F. **Deep Learning With Python**. Manning, 2018. 9 p.

DENG, J. et al. ImageNet: A Large-Scale Hierarchical Image Database. **CVPR09**, 2009.

HINTON, G. E.; OSINDERO, S.; TEH, Y.-W. A Fast Learning Algorithm for Deep Belief Nets. **Neural Computation**, p. 1527–1554, 2006. Disponível em: <http://www.cs.toronto.edu/~fritz/absps/ncfast.pdf>. Acesso em: 4 Outubro 2018.

HINTON, G. E.; SALAKHUTDINOV, R. R. Reducing the Dimensionality of Data with Neural Networks. **Science**, v. 313, p. 504-507, 2006. Disponível em: <https://www.cs.toronto.edu/~hinton/science.pdf>. Acesso em: 10 Outubro 2018.

HOWARD, A. G. et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. **arXiv preprint arXiv:1704.04861**, 2017.

HOWARD, A. G.; ZHU, M. MobileNets: Open-Source Models for Efficient On-Device Vision. **Google AI Blog**, 2017. Disponível em: <https://ai.googleblog.com/2017/06/mobilenets-open-source-models-for.html>. Acesso em: 3 Setembro 2018.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet Classification with Deep Convolutional Neural Networks. **Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems**, p. 1106-1114, 2012. Disponível em: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>. Acesso em: 4 Outubro 2018.

LECUN, Y. et al. Gradient-Based Learning Applied to Document Recognition. **Proceedings of the IEEE**, 1998. Disponível em: <http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf>. Acesso em: 3 Outubro 2018.

LISTA DE ESPÉCIES DA FLORA DO BRASIL. Portal Brasileiro de Dados Abertos, 2015. Disponível em: <http://dados.gov.br/dataset/floradobrasil>. Acesso em: 10 Agosto 2018.

MCCULLOCH, W. S.; PITTS, W. A Logical Calculus Of The Ideas Immanent in Nervous Activity. **Bulletin of Mathematical Biophysics**, v. 5, p. 115-133, 1943. Disponível em: <http://www.cs.cmu.edu/~epxing/Class/10715/reading/McCulloch.and.Pitts.pdf>. Acesso em: 15 Agosto 2018.

NILSBACK, M.-E.; ZISSERMAN, A. 102 Category Flower Dataset, 2008. Disponível em: <http://www.robots.ox.ac.uk/~vgg/data/flowers/102/>. Acesso em: 10 Agosto 2018.

OQUAB, M. et al. Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks. **Proceedings of the IEEE conference on computer vision and pattern recognition**, p. 1717-1724, 2014. Disponível em: <https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Oquab_Learning_and_Transferring_2014_CVPR_paper.pdf>. Acesso em: 7 Outubro 2018.

PATTANAYAK, S. **Pro Deep Learning with TensorFlow:** A Mathematical Approach to Advanced Artificial Intelligence in Python. Apress, 2017.

ROSENBLATT, F. The Perceptron: A Perceiving and Recognition Automation, 1957. Disponível em: <https://blogs.umass.edu/brain-wars/files/2016/03/rosenblatt-1957.pdf>. Acesso em: 28 Setembro 2018.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, v. 323, 1986. Disponível em: <https://www.iro.umontreal.ca/~vincentp/ift3395/lectures/backprop_old.pdf>.

SOKOLOVA, M.; LAPALME, G. A systematic analysis of performance measures for classification tasks. **Information Processing and Management**, v. 45, p. 427–437, 2009. Disponível em: <http://rali.iro.umontreal.ca/rali/sites/default/files/publis/SokolovaLapalme-JIPM09.pdf>. Acesso em: 20 Setembro 2018.

APÊNDICE A – Matriz de confusão da rede selecionada

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
A	58	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0
B	0	65	0	0	1	0	0	1	0	0	1	0	0	0	0	0
C	0	0	50	0	2	0	0	1	0	0	0	0	0	0	2	1
D	0	0	0	79	0	0	0	1	0	0	2	2	0	0	1	1
E	0	1	0	0	62	0	0	0	0	0	0	0	0	1	0	0
F	0	1	0	0	5	56	0	0	0	0	0	0	0	0	0	0
G	0	1	0	0	0	0	47	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	51	0	2	1	0	0	0	0	1
I	1	1	0	0	0	0	0	0	75	1	0	0	0	0	0	0
J	0	1	0	0	0	0	0	0	0	64	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	1	57	0	0	0	0	1
L	0	0	0	0	0	0	0	0	0	0	0	52	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	51	0	0	0
N	0	0	2	0	0	0	0	0	0	0	0	0	0	51	0	0
O	0	0	0	0	0	0	0	1	0	1	0	1	0	0	47	0
P	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	54

Legenda: A – *Bellis sp.* (Margarida); B – *Bougainvillea sp.* (Primavera); C – *Helianthus annuus* (Girassol); D – *Hibiscus sp.* (Hibisco); E – *Ixora coccínea* (Ixora); F – *Lantana camara* (Lantana); G – *Narcissus sp.* (Narciso); H – *Nerium oleander* (Oleandro); I – Nenhuma; J – *Plumbago auriculata* (Bela Emília); K – *Rhododendron simsii* (Azaleia); L – *Rosa sp.* (Rosa); M – *Strelitzia reginae* (Estrelítzia); N – *Taraxacum officinale* (Dente-de-leão); O – *Thunbergia alata* (Olho-Preto); P – *Viola x wittrockiana* (Amor-perfeito).