UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
UNIDADE ACADÊMICA DE SISTEMAS E COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MATHEUS GOMES MAIA

SMART GALLERY:

CLASSIFICAÇÃO E FILTRAGEM DE IMAGENS EM CELULARES

CAMPINA GRANDE - PB

2019

MATHEUS GOMES MAIA

SMART GALLERY:
CLASSIFICAÇÃO E FILTRAGEM DE IMAGENS EM CELULARES

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Professor Dr. Herman Martins Gomes.

CAMPINA GRANDE - PB

2019

M217s Maia, Matheus Gomes.
Smart Gallery : classificação e filtragem de imagens em celulares. / Matheus Gomes Maia. – 2019.

9 f.

Orientador: Prof. Dr. Herman Martins Gomes.
Trabalho de Conclusão de Curso - Artigo (Curso de Bacharelado em Ciência da Computação) - Universidade Federal de Campina Grande; Centro de Engenharia Elétrica e Informática.

1. Classificação de imagens. 2. Aplicativo mobile. 3. Redes neurais. 4. Galeria de fotos – classificação. I. Gomes, Herman Martins. II. Título.

CDU:004.6(045)

Elaboração da Ficha Catalográfica:

Johnny Rodrigues Barbosa
Bibliotecário-Documentalista
CRB-15/626

MATHEUS GOMES MAIA

SMART GALLERY:
CLASSIFICAÇÃO E FILTRAGEM DE IMAGENS EM CELULARES

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. Herman Martins Gomes
Orientador – UASC/CEEI/UFCG

Professor Dr. Thiago Emmanuel Pereira da Cunha Silva
Examinador – UASC/CEEI/UFCG

Professor Dr. Tiago Lima Massoni
Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 02 de julho 2019.

CAMPINA GRANDE - PB

Smart Gallery: Classificação e Filtragem de Imagens em Celulares

Matheus Gomes Maia
Departamento de Sistemas e Computação, Universidade Federal de Campina Grande
Campina Grande, Paraíba
matheus.maia@ccc.ufcg.edu.br

Orientador: Herman Martins Gomes, PhD
Departamento de Sistemas e Computação, Universidade Federal de Campina Grande
Campina Grande, Paraíba
hmg@dsc.ufcg.edu.br

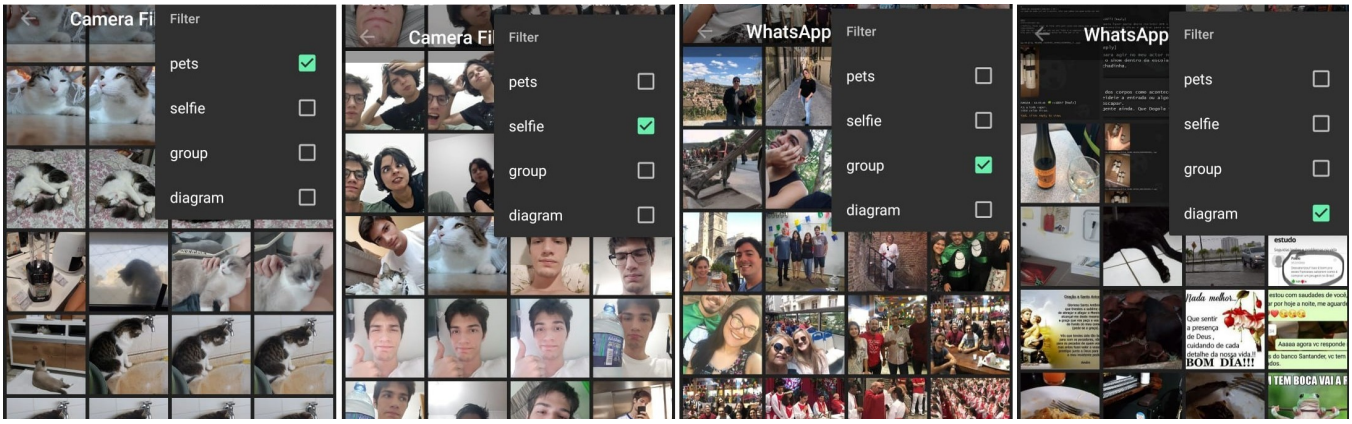

Figura 1: Resultado da classificação de álbuns pelo aplicativo Smart Gallery. Fonte: capturas de tela da aplicação no celular do próprio autor.

RESUMO

Atualmente, os celulares têm sido a ferramenta mais utilizada para captura e armazenamento de fotos. Estima-se que foram capturadas 1,2 trilhões de fotos em dispositivos móveis, em 2017. Tendo isso em vista, foi criada uma demanda por ferramentas de organização, armazenamento e proteção de fotos. Nesse contexto, o propósito deste trabalho é desenvolver um aplicativo que realize classificação de imagens na plataforma móvel para fins de organização. O aplicativo tem como funcionalidade principal filtrar as imagens de um álbum para serem exibidas apenas as fotos que se encaixarem nas classes de interesse do usuário, escolhidas dentre as classes disponíveis, como animais de estimação e *selfies*. Com as fotos filtradas, o usuário pode organizar, remover ou compartilhar as fotos. O aplicativo também foi concebido para proporcionar um sentimento de diversão e curiosidade, já que o resultado da classificação é imprevisível e existe um valor emocional atrelado às fotos pessoais. Um modelo de Redes Neurais Convolucional foi treinado para servir a classificação de imagens no cenário de galeria de fotos pessoais. Esse desafio é pouco explorado nas lojas de aplicativos e exige cuidado na criação da base de dados utilizada e no treinamento de um modelo de classificação satisfatório. Foi realizado um estudo de viabilidade com usuários, em que os pesquisados consideraram a funcionalidade de classificação de imagens do aplicativo útil e os resultados para a classificação das imagens satisfatórios.

Palavras Chave: aplicativo mobile, classificação de imagens, redes neurais **Repositório Estruturado:** https://github.com/matheusgmaia/Smart-Galley

1 INTRODUÇÃO

O volume de imagens capturadas aumentou significativamente na ultima década com a popularização dos celulares, fato apontado como a principal causa da queda de 84% nas vendas de câmeras digitais entre 2010 e 2019[6] e no aumento da quantidade de fotos capturadas nos últimos anos. Em 2017 estima-se que foram capturadas 1.2 trilhões de fotos contra 660 bilhões em 2013[5].

Nesse contexto, o propósito deste trabalho é desenvolver um aplicativo que realize a classificação de imagens na plataforma móvel, tornando-se mais acessível à maioria dos usuários. O aplicativo tem como funcionalidade principal filtrar as imagens de um álbum da galeria do celular para serem exibidas apenas as fotos cujo conteúdo se encaixem nas classes de interesse escolhidas pelo usuário, como animais de estimação e autorretratos (*selfies*).

Com as fotos filtradas, o usuário pode remover, compartilhar ou organizar em pastas uma seleção das fotos filtradas pela inteligência do aplicativo.

Uma funcionalidade similar pode ser encontrada, embora de difícil visualização, dentro do aplicativo Google Photos, onde é possível pesquisar por termos como "gato"ou "praia"na caixa de busca e então encontrar imagens relacionadas aos termos pesquisados. Diferentemente do aplicativo citado, no Smart Gallery as classes disponíveis são listadas ao usuário e disponibilizadas de forma *offline*.

Apesar de existirem grandes desafios resolvidos no contexto de classificação de imagem, o cenário de uma galeria de fotos pessoais é pouco explorado por aplicativos móveis, sendo o aplicativo do Google o único encontrado com uma funcionalidade semelhante na loja virtual de aplicativos para a plataforma Android, a Google Play.

A solução utilizada para a classificação das imagens é um modelo de Rede Neural Convolucional, treinado com milhares de imagens rotuladas com as classes de interesse disponíveis. Foram coletadas imagens pertencentes a quatro classes, para poupar um maior esforço na criação da base

de dados e concentrar o trabalho no treinamento do modelo neural e no desenvolvimento do aplicativo.

A Seção 2 inclui fundamentação teórica e explicação do problema. A Seção 3 contém uma visão geral da solução e principais decisões arquiteturais. A Seção 4 inclui imagens e demonstrações do sistema em uso. A Seção 5 apresenta e discute os resultados de um estudo de viabilidade com usuários. A Seção 6, por fim, resume o processo de desenvolvimento e as lições apreendidas durante o desenvolvimento, além de propor trabalhos futuros.

2 FUNDAMENTAÇÃO

O interesse pelo subcampo de Inteligência Artificial, Aprendizagem de Máquina, explodiu na última década. *Softwares* de detecção de *spam*, sistemas de recomendação, marcação em fotos de redes sociais, assistentes pessoais ativados por voz, carros autônomos, reconhecimento facial e muito outros tem sido desenvolvidos e estão atingindo níveis de estado da arte devido às recentes inovações no subcampo de Aprendizagem de Máquina.

Existem três principais abordagens para o Aprendizado de Máquina, sendo elas, aprendizado supervisionado, não supervisionado e por reforço, cujas abordagens se diferem pelo tipo de dados que utilizam para o aprendizado. Na abordagem supervisionada a máquina aprende a partir de exemplos rotulados, ao contrário da abordagem não supervisionada, que não necessita de rótulos; na abordagem por reforço a máquina aprende ao ser recompensada ou punida por suas decisões. Os tipos de aprendizado e algumas aplicações comumente resolvidas por cada tipo estão apresentadas na Figura 2.

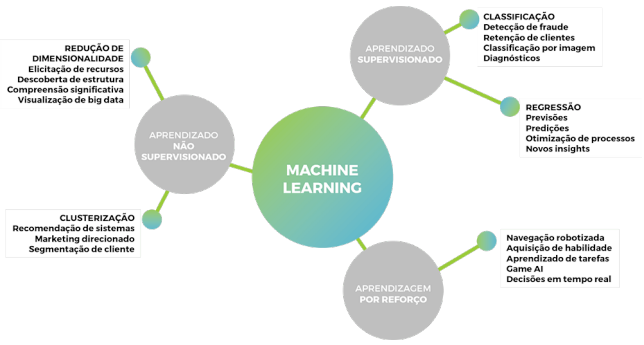

Figura 2: Tipos de abordagens para o Aprendizado de Máquina e aplicações. Fonte: Deal Lab

Existem algumas aplicações de Inteligência Artificial que são melhor resolvidas por uma abordagem específica de Aprendizagem de Máquina, enquanto que o problema de Classificação por Imagens, por exemplo, é melhor resolvido pela abordagem supervisionada.

Dentre os ramos existentes no contexto de Aprendizagem de Máquina, o que mais tem evoluído é o de Aprendizagem Profunda (Deep Learning), no qual soluções que utilizam Redes Neurais Profundas tem sido desenvolvidas com grande sucesso para resolver problemas em uma variedade de campos, como preprocessamento digital de imagem, visão computacional, processamento de linguagem natural, segurança e tantos outros.

O campo de visão computacional é um campo interdisciplinar que objetiva a obtenção de um entendimento de alto nível aos computadores a partir de imagens e vídeos. Tarefas de visão computacional incluem métodos para aquisição, processamento, análise e compreensão de imagens digitais para produzir informações numéricas ou simbólicas, na forma de decisões, por exemplo. Esse campo se beneficiou imensamente das recentes evoluções em redes neurais profundas, especialmente de uma classe de redes chamada redes neurais convolucionais, as quais tem o poder de extrair características locais das imagens e construir representações cada vez mais abstratas do conteúdo delas. Um modelo de rede neural convolucional já conseguiu superar um ser humano em um desafio de classificação de imagens, uma tarefa até então realizada por humanos de maneira inigualável[7].

Ao treinar uma rede neural, algumas técnicas podem ser utilizadas para acelerar o treinamento e obter uma melhor acurácia ou performance. Técnicas como transferência de conhecimento (*transfer learning*)[3] e aumento de dados (*data augmentation*)[4] foram utilizadas no treinamento da rede neural desenvolvida. Transferência de conhecimento se dá quando um modelo já treinado é utilizado em um contexto diferente, mas similar ao contexto original, como pontapé para o treinamento ou como extrator de características, permitindo uma maior eficiência para o processo de treinamento. Aumento de dados é uma técnica utilizada para expandir artificialmente o conjunto utilizado no aprendizado ao reinserir no conjunto exemplos disponíveis modificados. No contexto de imagens, mudança de brilho, contraste e orientação são algumas das muitas modificações possíveis ao realizar um aumento de dados. Uma maior quantidade de dados ajuda a aumentar o poder de generalização do modelo e, portanto, sua robustez no momento de uso.

3 ARQUITETURA E PROJETO DA SOLUÇÃO

Nesta seção serão apresentadas as decisões de projeto tomadas durante o desenvolvimento do aplicativo, sendo divididas entre as pertinentes ao treinamento do modelo e as pertinentes ao desenvolvimento do aplicativo que utiliza o modelo.

A Figura 3 apresenta um diagrama simplificado da arquitetura da solução. O desafio central da aplicação, classificação de imagens, é um desafio de visão computacional e aprendizado de máquina, comumente resolvido utilizando a abordagem supervisionada. Portanto, um conjunto de exemplos rotulados foi coletado e então uma Rede Neural Convolucional foi treinada para realizar a tarefa de classificar imagens dentre as classes disponíveis. Foi utilizada a biblioteca TensorFlow na linguagem python para o treinamento da rede. O aplicativo que utiliza a rede treinada para permitir as funcionalidades de filtrar imagens foi desenvolvido para a plataforma Android, utilizando o seu SDK na linguagem Java. A API Glide, utilizada para recuperar as imagens do sistema de arquivos do celular e a API TensorFlow Lite, utilizada para permitir a utilização da rede no ambiente móvel, são as duas principais bibliotecas utilizadas.

3.1 Modelo de Classificação

As duas principais tarefas para se obter um modelo de classificação são a confecção da base de dados seguida pelo treinamento da rede. Nessa seção serão discutidas essas duas tarefas.

3.1.1 Base de Dados. Para a base de dados utilizada foram coletadas imagens para cada uma das quatro classes de interesse escolhidas, sendo essas: animais de estimação (*Pets*), grupos de pessoas (*Groups*), auto-retratos (*Selfies*) e diagramas (*Diagrams*), que são imagens não fotográficas, como tirinhas e montagens frequentemente compartilhadas nas redes sociais.

Essas classes foram escolhidas pela facilidade em se obter milhares de fotos de exemplo e por serem imagens comumente encontradas em uma galeria de fotos pessoais e por serem úteis para funcionalidade de organização da galeria pessoal, na opinião do autor.

Informações sobre a quantidade de imagens disponíveis para cada classe e sobre a origem das imagens estão descritas na Tabela 1.

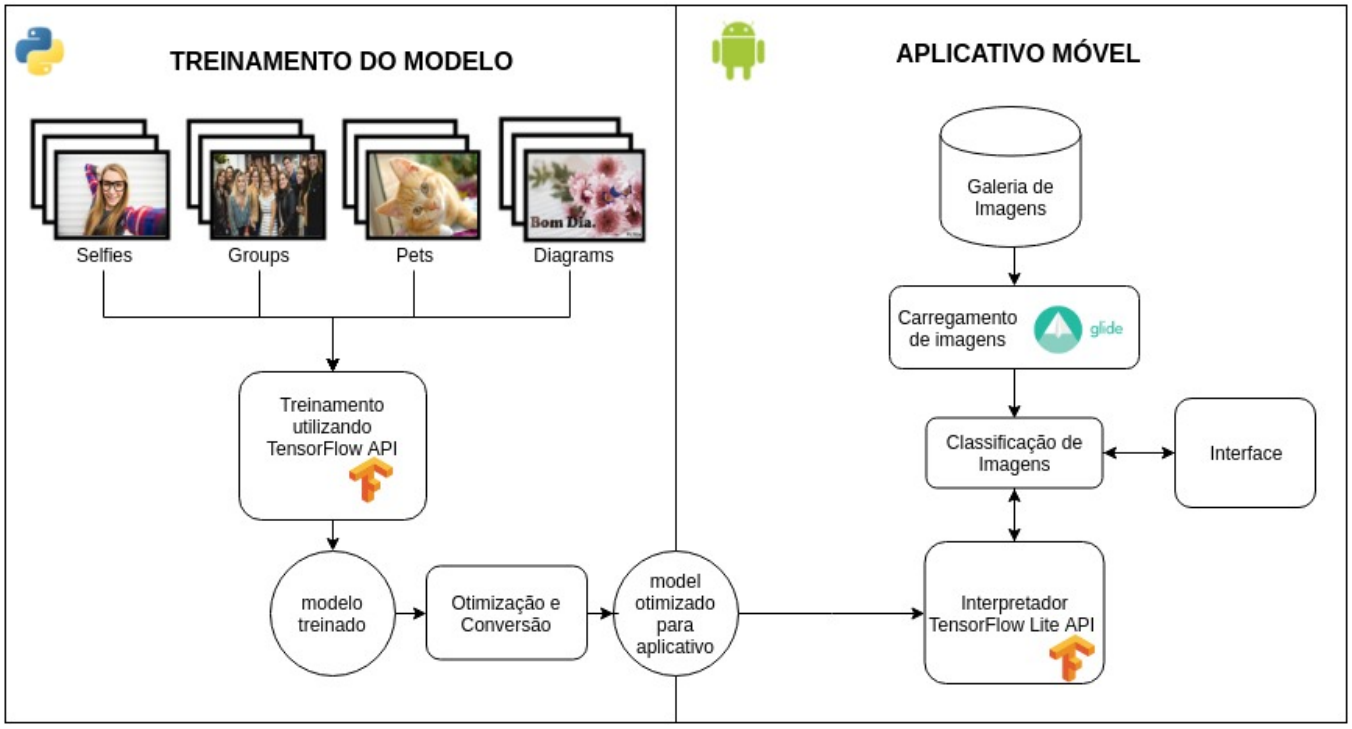

Figura 3: Diagrama simplificado da arquitetura da solução. Fonte: próprio autor.

Tabela 1: Classes de interesse, quantidade de exemplos coletados e origem dos exemplos

Classe	Nº de Exemplos Coletados	Origem
Selfies	10.000	UCF Datasets [2]
Pets	25.000	Kaggle Dogs vs Cats Dataset
Groups	5.075	Cornell University [1]
Diagrams	8.139	*BING Images API*

Como podemos observar na Tabela 1, cada classe tem uma origem diferente para os exemplos utilizados no treinamento. A classe *Pets*, formada por fotos de gatos e cachorros, está disponível no site de desafios para cientistas de dados, Kaggle Dogs vs Cats Dataset. As imagens para as classes *Selfies* e *Groups* foram montadas a partir de fotos disponíveis na internet por universidades americanas e estão disponíveis nos sites das instituições[2][1]. A classe *Diagrams* foi concebida para filtrar apenas imagens não fotográficas, como montagens e tirinhas, e foi criada para ser utilizada apenas nesse treinamento. Algumas ferramentas foram utilizadas para reunir exemplos para a classe *Diagrams*, como a API do *BING Images*, onde se pode pesquisar imagens indexadas pela ferramenta de busca de imagens e filtrar os resultados, tendo sido utilizadas apenas as imagens que a ferramenta classificou como *ClipArt*.

Para o treinamento, foram utilizadas dez mil imagens de cada classe. Os conjuntos *Groups* e *Diagrams* passaram por uma etapa de aumento de dados e o conjunto *Pets* foi reduzido. Para o aumento de dados, a biblioteca Augmentor foi utilizada, de modo que, modificações no brilho e contraste e transformações de rotação, espelhamento e ampliação foram realizadas de maneira randômica, com o suporte da ferramenta. Na Figura 4, é apresentado um exemplo de aumento de dados realizado em uma imagem da classe *Groups*.

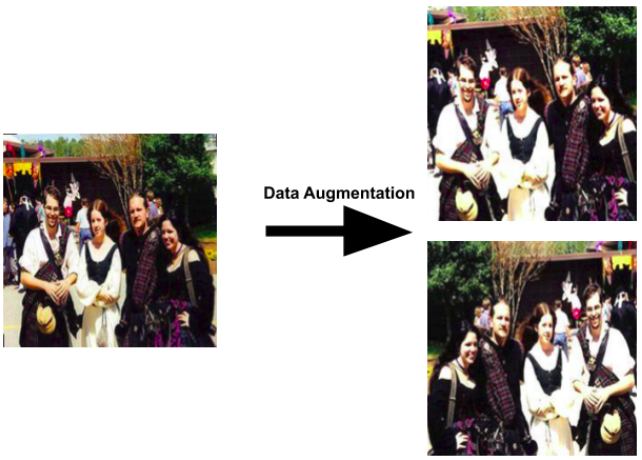

Figura 4: Exemplo de aumento de dados onde modificações de espelhamento, inclinação e brilho foram realizadas. Fonte: próprio autor.

3.1.2 Treinamento. O treinamento foi realizado utilizando a biblioteca Tensorflow, uma biblioteca de código aberto para computação numérica que utiliza fluxos de dados em um grafo, desenvolvida pelo Google, para projetos de Aprendizagem de Máquina e Aprendizagem Profunda. A linguagem utilizada para acessar as funções da biblioteca foi a linguagem Python. Os

tutoriais e documentação oficiais, disponíveis no site da biblioteca, serviram de referência para o treinamento realizado.

A arquitetura da rede utilizada foi a MobileNetV2[8], uma arquitetura proposta para funcionar em plataformas móveis e embarcados, as quais possuem recursos limitados. Como ponto de partida para o treinamento, foi utilizado um modelo treinado da arquitetura MobileNetV2, disponível no site do Tensorflow, para o desafio ILSVRC-2012-CLS[7], um desafio de classificação de imagens para mil classes utilizando cento e cinquenta mil imagens. Reutilizar modelos treinados anteriormente como ponto de partida para acelerar o treinamento de novos modelos, usualmente em problemas correlatos, chama-se transferência de conhecimento. Esta técnica possibilita uma redução no tempo de treinamento e na quantidade de imagens exemplo necessárias.

A rede MobileNet foi proposta acompanhada de dois parâmetros que controlam a sua arquitetura, chamados *trade-off hyper parameters*. O parâmetro *Width Multiplier* controla a largura da rede e varia entre 1.4 e 0.35. O parâmetro *Resolution* controla a dimensão das imagens utilizadas e varia entre 224 e 96. Os valores desses parâmetros influenciam diretamente a latência da rede, como pode ser observado na Tabela 2.

Tabela 2: Variações da arquitetura MobileNetV2, acurácia no desafio ImageNet Challenge: ILSVRC 2012 e latência da arquitetura no celular Pixel 1. Fonte: Github TFModels

Arquitetura da rede	Acurácia ImageNet (TOP 5)	Latencia(ms)
mobilenet_v2_1.4_224	92.5%	138.0
mobilenet_v2_1.3_224	92.1%	123.0
mobilenet_v2_1.0_224	91.0%	73.8
mobilenet_v2_0.75_224	89.6%	55.8
mobilenet_v2_0.50_224	86.4%	28.7
mobilenet_v2_0.35_224	82.4%	19.7
mobilenet_v2_1.0_224	91.0%	73.8
mobilenet_v2_1.0_196	90.1%	55.1
mobilenet_v2_1.0_160	89.0%	42.2
mobilenet_v2_1.0_128	86.9%	27.6
mobilenet_v2_1.0_96	83.2%	17.6
mobilenet_v2_0.75_160	87.3%	30.4

Como é perceptível na Tabela 2, existe um custo de latência ao optar por uma rede que proporcione uma maior acurácia. A variação da arquitetura utilizada para o treinamento foi a MobilenetV2_0.75_160, por ter parâmetros medianos e baixa latência.

O conjunto de imagens disponíveis foi dividido entre treino, validação e teste na proporção 80%, 10% e 10%, respectivamente. Nas Figuras 5 e 6, estão apresentados os gráficos de acurácia e erro (*Loss*) para o treinamento da Rede Neural no conjunto de treino, em laranja, e validação, em azul.

A acurácia no conjunto de treino informa a porcentagem de acertos na classificação, relativa ao grupo de imagens que estão sendo utilizadas no treinamento naquele instante (*current training batch*). A acurácia no conjunto de validação informa a porcentagem de acerto em todas as imagens do conjunto de validação, sendo calculada a cada cem épocas. O erro (*Loss*) da classificação no conjunto de treino e validação é obtido a partir da seguinte fórmula:

$$Loss = -\sum_{c=0}^{M} y_{o,c} \log(p_{o,c}) \quad (1)$$

Onde,

- M é o numero da classe (*selfies*, *pets*, *groups* e *Diagrams*);
- log é o log natural;
- y é um indicador binário (0 ou 1) que retorna 1 se a classificação c foi correta para a observação o; e
- p é a probabilidade da classe c para a observação o

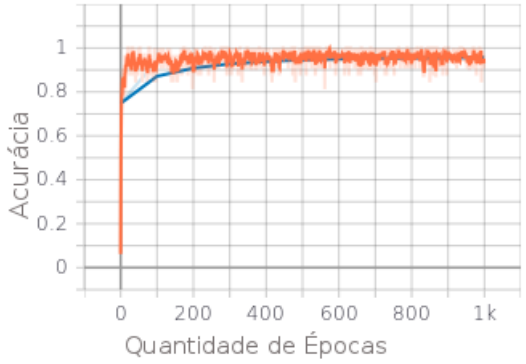

Figura 5: Gráfico da acurácia ao decorrer do treinamento. Acurácia no conjunto de treino em laranja e no conjunto de validação, em azul. Fonte: próprio autor, com o auxílio da ferramenta *TensorBoard*.

Figura 6: Gráfico do (*Loss*) ao decorrer do treinamento. (*Loss*) no conjunto de treino em laranja e no conjunto de validação, em azul. Fonte: próprio autor, com o auxílio da ferramenta *TensorBoard*.

Podemos perceber que o treinamento atingiu, já nas primeiras 500 épocas, uma acurácia alta e um *Loss* baixo. A acurácia medida na época final no conjunto de treino ficou em 93,8%, no conjunto de validação 95,9% e no conjunto de teste em 96,4%. A rede chegou a 100% de acurácia de treino em algum ponto do treinamento. Os *logs* completos e evidências do treinamento estão disponíveis no repositório do aplicativo.

Para que o modelo treinado no computador fique interpretável pela API utilizada na plataforma móvel para predição, TensorFlow Lite, faz-se necessária uma conversão para o formato apropriado. Além da conversão para o formato apropriado, algumas otimizações foram feitas para melhorar a performance do modelo na plataforma Android, como quantização, assim como é recomendado pelos tutoriais oficiais.

Neste trabalho, para treinamento do modelo neural, foi utilizado um computador com a seguinte configuração: processador Intel Core i7 8700 - CPU 3.20GHz, com 32GB RAM DDR4, Nvidia GTX 1080 8GB, Hard Disk de 1TB e Ubuntu 16.04 x64.

3.2 Aplicativo Móvel

O primeiro passo para a implementação do aplicativo foi encontrar uma solução de galeria de fotos com código aberto para utilizar como base. O aplicativo escolhido chama-se Camera Roll e utiliza a biblioteca Glide API para recuperar as imagens do sistema de arquivos do dispositivo e tem funcionalidades que interessam ao aplicativo Smart Gallery, como deletar imagens e criar um álbum a partir de uma seleção de imagens. A biblioteca Glide API, herdada do código base, e a biblioteca TensorFlow Lite são as principais bibliotecas utilizadas pelo aplicativo.

Funcionalidades de classificação de imagem foram adicionadas ao aplicativo base, enquanto outras foram removidas como personalizar tema do aplicativo e explorar sistema de arquivos, para simplificar o código e evidenciar a classificação de imagens.

4 SISTEMA EM USO

Nessa seção são apresentadas algumas das funcionalidades principais do aplicativo por meio de captura de telas.

Ao abrir o aplicativo o usuário é apresentado a todos os álbuns que estão em seu celular, como ilustrado na Figura 7.

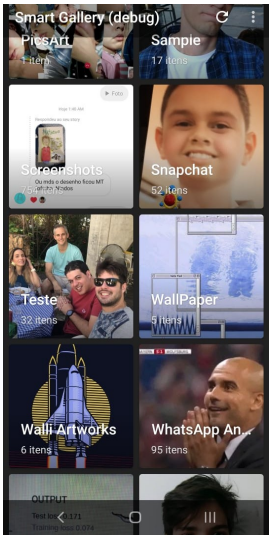

Figura 7: Tela inicial do aplicativo onde os álbuns do usuário são exibidos. Fonte: captura de tela da aplicação no celular do próprio autor.

Ao selecionar um álbum, o usuário pode ir nas opções do canto da tela e selecionar a opção *filter*, onde então estarão listados os filtros disponíveis. Ao selecionar um filtro as imagens são carregadas aos poucos, mas apenas as que se encaixam no filtro escolhido são exibidas, como ilustrado na Figura 8 , em que o filtro *diagrams* está selecionado.

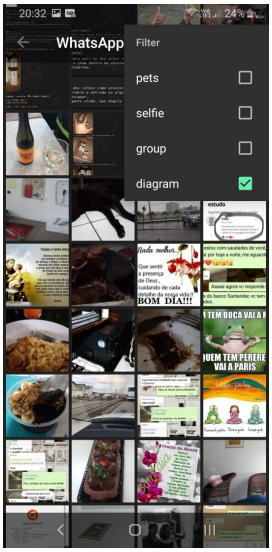

Figura 8: Exemplos de álbum filtrado para o filtro Diagrams. Podemos perceber alguns erros. Fonte: captura de tela da aplicação no celular do próprio autor.

Podemos perceber na Figura 8 que o filtro falhou em algumas imagens, que são imagens fotográficas mas que foram classificadas como *diagrams* de maneira equivocada.

Com as imagens filtradas, o usuário pode selecionar uma grupo de imagens para então compartilhar, copiar, mover ou excluir essa seleção, como ilustrado na Figura 9.

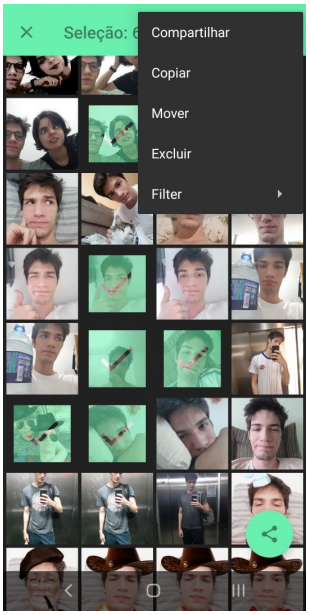

Figura 9: Seleção de imagens que a princípio foram filtradas como *selfies* pelo aplicativo. É possível ver opções do que fazer com a seleção de imagens. Fonte: captura de tela da aplicação no celular do próprio autor.

5 ESTUDO COM USUÁRIOS

O aplicativo Smart Gallery foi testado por 7 usuários que, após a utilização, responderam a um rápido questionário.

O questionário consiste em três perguntas. Primeiramente foi pedido para os usuários avaliarem em uma escala de 1 a 5 o quanto o Smart Gallery pode ser útil na organização da sua galeria pessoal. Em seguida, foi pedido para avaliarem em uma escala de 1 a 5 a qualidade dos resultados dos filtros de imagens. Por último, foram solicitadas sugestões de melhorias ou comentários em relação à interface, filtragem de imagens e utilidade, caso pertinentes.

As duas primeiras questões foram baseadas na escala de Likert, com 5 níveis, onde 1 corresponde a "Muito insatisfeito", 2 a "Insatisfeito", 3 a "Neutro", 4 a "Satisfeito"e 5 a "Muito Satisfeito".

A Tabela 3 apresenta os resultados.

Tabela 3: Respostas dos usuários pesquisados sobre o aplicativo Smart Gallery e suas funcionalidades.

Usuário	Utilidade	Qualidades dos Resultados
Usuário 1	5	3
Usuário 2	4	3
Usuário 3	4	4
Usuário 4	5	4
Usuário 5	4	3
Usuário 6	4	4
Usuário 7	4	3
Média	4,28	3,42
Desvio Padrão	0,48	0,53

Os usuários em geral acharam as funcionalidades do aplicativo úteis. A qualidade dos resultados foi avaliada com uma média de 3.42, o que é esperado devido a enorme variabilidade de imagens encontrada durante o uso.

Algumas sugestões coletadas pela terceira pergunta estão listadas a seguir:

- "O filtro para pets, na minha opinião, precisa ser melhorado. Ele retornou diversas imagens que não condizem com o filtro ";
- "Acho que se houver uma espécie de tutorial no início pode ajudar a fixar melhor como executar as funcionalidades ";
- "Mais classes para organizar a galeria e busca por todos os álbuns, além de melhoria na detecção "; e
- "A interface precisa melhorar ao mostrar as informações ao usuário, por exemplo, mostrar se a filtragem já foi concluída. E permitir escolher várias filtragens antes de fechar o menu de escolha ".

Por se tratar de um produto ainda inacabado, algumas das sugestões coletadas no formulário relatam problemas com a interface e com alguns resultados. A maioria das sugestões apontadas, referentes à interface, são pontuais e de fácil resolução mediante pequenas modificações. Melhorias na classificação podem ser alcançadas com mais experimentos e aprimoramentos na base de dados.

O aplicativo funcionou dentro do esperado em todos os celulares testados durante o desenvolvimento e estudo com usuários, a saber:

- Samsung A8-Android 9.0;
- Moto G4 Plus-Android 7.0;
- Xiaomi MiA2-Android 9.0;
- Samsung S9-Android 9.0;
- Xiaomi Mi 8 Lite- Android 9.0;
- Moto G7 Play-Android 9.0;
- Moto X4-Android 9.0; e
- Sansung S7-Android 8.0;

6 EXPERIÊNCIA E LIÇÕES APREENDIDAS

Sobre o treinamento do modelo de classificação, o maior desafio foi coletar uma base de dados com milhares de fotos para cada classe e converter o modelo treinado para o formato que pode ser lido pela biblioteca TensorFlow Lite no aplicativo Android. Os tutoriais e documentação disponíveis no site oficial foram os guias nesta tarefa. O treinamento foi refeito algumas vezes com arquiteturas e bases diferentes, tendo em vista a obtenção de resultados pouco satisfatórios nos primeiros treinamentos.

No desenvolvimento do aplicativo, o primeiro passo foi escolher um código aberto para servir de ponto de partida. O código escolhido foi encontrado com a busca avançada do GitHub e cumpre os requisitos esperados. Com o código definido, as funcionalidades de classificação de imagens foram adicionadas ao aplicativo original e funcionalidades herdadas que não faziam sentido para a aplicação foram removidas. Mudar a execução do aplicativo ao carregar as imagens da memória e lidar com problemas de concorrência e de performance foram os maiores desafios.

Existe uma lista de aprimoramentos que poder servir como indicadores para trabalhos futuros, a saber:

- Aperfeiçoamentos na interface e performance para melhorar a experiência do usuário.
- Experimentos com diferentes variações da arquitetura do modelo de classificação;
- Análise do impacto da adição de mais classes;
- Fazer um modelo para cada classe e analisar o impacto de múltiplos modelos na performance e acurácia;
- Criar um sistema de *feedback* para que o usuário ajude a rede a aprender; e
- Criar uma loja virtual de modelos que possam ser baixados para que sejam utilizados de maneira offline.

AGRADECIMENTOS

Agradeço aos meus familiares, professores e colegas que me ajudaram imensamente durante todo o período em que estive na graduação. Agradeço ao professor Herman pela orientação e paciência. Por fim, agradeço especialmente à minha mãe, por sempre me apoiar, me encorajar e ser o meu modelo de vida.

REFERÊNCIAS

[1] A. Gallagher and T. Chen. 2009. Understanding Images of Groups of People. In *Proc. CVPR*.

[2] Mahdi M Kalayeh, Misrak Seifu, Wesna LaLanne, and Mubarak Shah. 2015. How to take a good selfie?. In *Proceedings of the 23rd ACM international conference on Multimedia*. ACM, 923–926.

[3] Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2009), 1345–1359.

[4] Luis Perez and Jason Wang. 2017. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. *CoRR* abs/1712.04621 (2017). arXiv:1712.04621 http://arxiv.org/abs/1712.04621

[5] Felix Richter. 2017. Smartphones Cause Photography Boom. Retrieved May 20, 2019 from https://www.statista.com/chart/10913/number-of-photos-taken-worldwide/

[6] Felix Richter. 2019. Digital Camera Sales Dropped 84% Since 2010. Retrieved May 22, 2019 from https://www.statista.com/chart/5782/digital-camera-shipments/

[7] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vision* 115, 3 (Dec. 2015), 211–252. https://doi.org/10.1007/s11263-015-0816-y

[8] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation. *CoRR* abs/1801.04381 (2018). arXiv:1801.04381 http://arxiv.org/abs/1801.04381