UPF UNIVERSIDADE DE PASSO FUNDO | EDITORA

RBCA
ISSN 2176-6649

Revista Brasileira de Computação Aplicada, Novembro, 2022

DOI: 10.5335/rbca.v14i3.13058
Vol. 14, Nº 3, pp. 1–16
Homepage: seer.upf.br/index.php/rbca/index

ARTIGO ORIGINAL

Aplicativo para auxiliar pessoas com deficiência visual no reconhecimento de cédulas de dinheiro em Real com a técnica de Redes Neurais Artificiais

Application to assist visually impaired people in the recognition of Real money bills with Artificial Neural Networks technique

Fabrício Samuel Sausen[1], and Rejane Frozza[1,2]

[1]Universidade de Santa Cruz do Sul, [1]Departamento de Engenharias, Arquitetura e Computação, [2]Programa de Pós-graduação em Sistemas e Processos Industriais

*fabriciosausen@mx2.unisc.br; frozza@unisc.br

Recebido: 15/10/2021. **Revisado:** 26/07/2022. **Aceito:** 26/09/2022.

Resumo

A deficiência visual afeta cerca de 3,4% da população brasileira que declarou ter algum tipo de deficiência, no censo do IBGE de 2010, ou seja, cerca de 443 mil pessoas. Dentre os diversos problemas enfrentados por elas, destaca-se o uso do dinheiro, para o qual dependem do auxílio de pessoas conhecidas para realizar a identificação das cédulas. Neste sentido, este trabalho teve como objetivo desenvolver um aplicativo para *smartphones* que as auxilie no reconhecimento de cédulas de dinheiro em Real. Para realizar o reconhecimento das imagens das cédulas de dinheiro em Real foram utilizadas as Redes Neurais Artificiais, que vêm ganhando visibilidade quando se trata de reconhecimento de padrões em imagens. Como resultados obtidos, destaca-se que o modelo reconhecedor de cédulas de dinheiro teve uma assertividade de 95% sobre o *dataset* de teste. Um processo de avaliação do aplicativo desenvolvido foi realizado com cinco usuários com deficiência visual, que utilizaram-no e responderam questões a fim de mensurar o seu nível de satisfação em relação ao uso do aplicativo.

Palavras-Chave: Cédulas de dinheiro; deficiência visual; reconhecimento de imagem; redes neurais artificiais.

Abstract

Visual impairment affects about 3.4% of the Brazilian population which declared having some type of disability, in the 2010 IBGE census, that is, about 443 thousand people. Among the various problems faced by them, the use of money stands out, for which they depend on the help of people they know to carry out the identification of the money bills. In this sense, this work aimed to develop an application for smartphones that helps them to recognize Brazilian's money bills. Artificial Neural Networks were used for the recognition process, which have been gaining visibility when it comes to image pattern recognition. As results, it is highlighted that the Brazilian's money bill recognition model had an accuracy of 95% on the test dataset. An evaluation process of the developed application was carried out with five visually impaired users, who used it and answered questions in order to measure their level of satisfaction regarding the use of the mobile application.

Keywords: Banknotes; visual impairment; image recognition; artificial neural networks.

1 Introdução

O número de pessoas com algum tipo de deficiência no Brasil é grande. Segundo dados do Censo Demográfico de 2010, coletados pelo Instituto Brasileiro de Geografia e Estatística - IBGE, do total da população brasileira, em média 6,7% ou 12,7 milhões de pessoas declaram ter algum tipo de deficiência. Dentre as deficiências declaradas no censo, a mais comum mostrou-se ser a deficiência visual, atingindo 3,4% da população que a declarou, ou seja, cerca de 443.454 pessoas (IBGE, 2018).

Dentre as dificuldades enfrentadas por pessoas com deficiência visual, destaca-se a dependência. Pelo fato de não enxergarem, a locomoção, o trabalho e a identificação de objetos são atividades complicadas de serem realizadas de forma independente. Esta última, a identificação de objetos, engloba diversos aspectos do cotidiano de uma pessoa inserida em sociedade, como por exemplo, a identificação e uso de dinheiro.

Em muitos casos, a identificação e, consequentemente, o uso do dinheiro ainda é um desafio para pessoas com deficiência visual, pois dependem de auxílio de pessoas conhecidas para sua contagem e separação. Na rua, dependem da boa vontade e das virtudes de cada cidadão para não serem enganados, ao receber o troco em estabelecimentos comerciais, feiras, dentre outros.

Visando tornar mais independente a vida de pessoas com deficiência visual, neste trabalho foi desenvolvido um aplicativo capaz de reconhecer cédulas de dinheiro em Real, de forma automática, através de um *smartphone*. Para tornar isto possível, foram estudados e aplicados métodos de Aprendizado de Máquina (AM), mais especificamente as técnicas de Redes Neurais Artificiais (RNAs), como as Redes Neurais Convolucionais (RNCs).

O artigo está organizado nas seguintes seções: a seção 2 apresenta a fundamentação; a seção 3 destaca informações do *dataset* utilizado; a seção 4 apresenta o reconhecedor de cédulas de dinheiro em Real, bem como seus resultados; a seção 5 apresenta informações sobre o aplicativo desenvolvido e seus resultados obtidos; por fim, a seção 6 apresenta a conclusão do artigo.

2 Fundamentação

Os assuntos relacionados a esta pesquisa são apresentados a seguir: pessoas com deficiência visual, aprendizado de máquina e redes neurais artificiais.

2.1 Pessoas com deficiência visual

De modo geral, a deficiência visual pode ser dividida em dois grupos: a cegueira total e a cegueira parcial. Segundo CONDE (2017), a cegueira total, também conhecida como amaurose, pressupõe completa perda de visão, ou seja, nem a percepção luminosa está presente, necessitando, por exemplo, de instrução em Braile (sistema de escrita por pontos em relevo). Já na cegueira parcial, também conhecida como baixa visão, os indivíduos são capazes de contar dedos a curta distância, perceber vultos e até mesmo ler impressos ampliados ou com auxílio de potentes recursos ópticos.

A deficiência visual, dependendo de seu nível, pode dificultar ou até impedir a realização de diferentes tarefas no dia a dia de uma pessoa, desde a escolha de uma roupa até a utilização de serviços em estabelecimentos, como em bancos. A dependência acaba, muitas vezes, sendo constante na vida destas pessoas e, assim, por falta de tecnologias e recursos facilitadores, acabam sendo, de certa forma, excluídas da vida em sociedade, mesmo que de forma não voluntária.

Como uma sociedade democrática, na qual todos devemos ter os mesmos direitos e deveres como cidadãos, faz-se necessário facilitar alguns aspectos da rotina de pessoas com deficiência, para que possam realizar tarefas em seu cotidiano e assim serem mais independentes, podendo exercer seus direitos e deveres como cidadãos. Assim, um conceito importante deve ser levantado, o de Tecnologia Assistiva ou ajuda técnica, definido pela Lei Nº 13.146 em seu Art. 3º, Item III, como:

> ... produtos, equipamentos, dispositivos, recursos, metodologias, estratégias, práticas e serviços que objetivem promover a funcionalidade, relacionada à atividade e à participação da pessoa com deficiência ou com mobilidade reduzida, visando à sua autonomia, independência, qualidade de vida e inclusão social (BRASIL, 2015).

A partir desta definição, nota-se a abrangência de recursos facilitadores que podem ser desenvolvidos para auxílio na promoção da funcionalidade de pessoas com deficiência, ou seja, um suporte que torne possível ou melhore a realização de uma tarefa, que sem a tecnologia assistiva, seria impraticável. Portanto, a tecnologia assistiva, potencializa a independência, melhora a qualidade de vida e proporciona a inclusão social de pessoas com deficiência.

2.2 Aprendizado de Máquina

O Aprendizado de Máquina (AM) é uma vertente da área de Inteligência Artificial (IA), que procura, através de dados de entrada, desenvolver algoritmos capazes de aprender com eles e assim auxiliar na tomada de decisões, como: reconhecimento de padrões, previsão e detecção de anomalias/fraudes. Segundo Rezende (2003), "Aprendizado de Máquina é uma área de IA cujo objetivo é o desenvolvimento de técnicas computacionais sobre o aprendizado bem como a construção de sistemas capazes de adquirir conhecimento de forma automática".

O Aprendizado de Máquina pode ser dividido em 5 tipos: i) Aprendizado Supervisionado; ii) Aprendizado Não-Supervisionado; iii) Aprendizado Semi-Supervisionado; iv) Aprendizado por Reforço; v) Aprendizado Profundo (*Deep Learning*). A seguir será brevemente explicado o i) e v), por fazerem parte da pesquisa desenvolvida.

2.2.1 *Aprendizado Supervisionado*

O Aprendizado Supervisionado é um método que necessita de dados de entrada (exemplos) previamente rotulados, os quais são utilizados para extração de conhecimento durante o processo de aprendizado (treinamento) de um modelo preditivo, capaz de classificar novos exemplos nunca antes vistos. No Aprendizado Supervisionado, os exem-

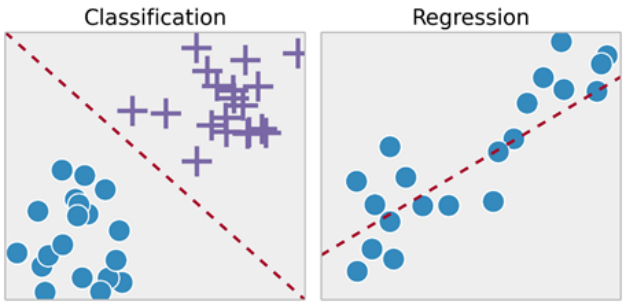

Figura 1: Exemplo de Classificação e Regressão

plos são rotulados com valores contínuos (por exemplo, números reais) ou categóricos (por exemplo, sim ou não - conjuntos finitos). Quando há exemplos com valores contínuos, a tarefa de aprendizado é chamada de Regressão e, quando há exemplos com valores categóricos, a tarefa de aprendizado é chamada de Classificação Pila (2001).

Um exemplo de aplicação da Classificação é: dado atributos relativos à imagem de uma pessoa, o modelo deve predizer se a pessoa é do sexo masculino ou feminino. Já para a regressão, não se prediz uma classe específica, mas sim um valor contínuo, sendo geralmente aplicado em perguntas do tipo: "quanto vai ser vendido no próximo mês?", dado um histórico de valores vendidos em meses anteriores. A Fig. 1 exemplifica diferença entre estas duas subcategorias de Aprendizado Supervisionado, pois se observa a clara separação em grupos na Classificação, e uma previsão do valor contínuo na Regressão.

2.2.2 *Deep Learning*

O Aprendizado Profundo (*Deep Learning*), é um método de aprendizado de máquina baseado em representações de dados, o qual se constitui da utilização de várias camadas de neurônios de processamento (daí o termo *Deep*), podendo utilizar algoritmos de aprendizagem supervisionada e não-supervisionada nas diferentes camadas. De acordo com Oppermann (2019), as camadas são capazes de aprender representações da entrada por conta própria, sem a necessidade da extrações de características (feature extraction) de forma manual, como rodas, formato e tamanho de um carro, por exemplo.

A cada camada, uma representação cada vez mais abstrata e complexa é reconhecida. Nielsen (2019) destaca que a primeira camada pode reconhecer as bordas, os neurônios da segunda camada podem aprender a reconhecer padrões mais complexos, como triângulos ou retângulos, construídos pelas bordas, e assim sucessivamente.

Um dos tipos mais populares de *Deep Learning* é conhecido como Redes Neurais Convolucionais (RNC), do inglês *Convolutional Neural Networks* (CNN). Segundo MATHWORKS (2021), uma RNC convolve características aprendidas com os dados de entrada, utilizando camadas de convolução 2D (de duas dimensões), tornando esse tipo de arquitetura de rede adequada para o processamento de dados 2D, como imagens.

A Fig. 2 apresenta um exemplo abstrato de *Deep Learning* através de Rede Neural Convolucional. Basicamente, a entrada da rede é representada por uma imagem de carro, a qual é passada pela RNC, composta por diversas camadas de convoluções, resultando em uma última camada de classificação do tipo de veículo.

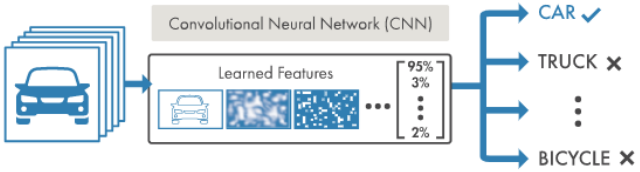

Figura 2: Exemplo de Deep Learning com RNC

2.3 Redes Neurais Artificiais

De acordo com Haykin (1999), uma Rede Neural pode ser vista como um processador distribuído massivamente paralelo feito de simples unidades de processamento, as quais possuem uma propensão natural para guardar conhecimento de experiências e tornar isso disponível para uso. Neste sentido, a RNA se assemelha com o cérebro humano em dois aspectos:

i. Conhecimento é adquirido pela rede através do processo de aprendizado.
ii. Forças de conexão entre neurônios, conhecidas como pesos sinápticos, são usados para guardar o conhecimento adquirido.

As unidades de processamento, ou neurônios, de uma RNA, são compostos basicamente por 3 elementos, sendo eles: (i) Conjunto de sinapses ou ligações, que são constituídas do valor do peso sináptico; (ii) Função de soma, que é responsável por somar a multiplicação dos valores de entrada com os pesos associados aos neurônios; (iii) Função de ativação, que é responsável por limitar a amplitude do valor atribuído a um neurônio (Haykin, 1999)).

Na Fig. 3, tem-se a representação de um neurônio artificial, sendo possível visualizar os elementos que o constituem, onde: *Input* são os valores de entrada, *Weights* são os valores dos pesos sinápticos, *Net input function* é a Função de Soma, *Activation function* é a Função de Ativação e, por fim, há o *output* que representa o valor de saída do neurônio.

A RNA é formada pela conexão de diversos neurônios artificiais, assim como demonstrado na Fig. 3, portanto é importante ressaltar que os cálculos referentes à Função de Soma e Função de Ativação ocorrem em cada um dos neurônios que compõem a RNA, podendo a saída (valor de ativação) de um neurônio servir de entrada para outro interligado a ele.

Na Fig. 4, o "x" representa o conjunto de dados de entrada, "W" e "b" representam, respectivamente, os valores de pesos sinápticos e bias, "a" representa o valor de ativação para os neurônios da camada oculta e "y" o valor estimado do neurônio da camada de saída (Ganssle, 2018). O bias ou viés é uma constante que ajuda o modelo de maneira que ele possa se ajustar melhor aos dados fornecidos,

ajudando a controlar o valor no qual a função de ativação será acionada (GEEKSFORGEEKS, 2019).

O processo de aprendizado, ou algoritmo de aprendizado de uma RNA possui a função de ajustar os pesos sinápticos, a fim de que, após um número finito de iterações, a rede tenha desemepenho o mais próximo do objetivo proposto. Além disso, de acordo com Rezende (2003), considera-se que o processo de aprendizado tem como característica a ocorrência de estímulo da rede pelo meio externo através da apresentação do conjunto de dados.

O conjunto de dados, no caso deste trabalho, é um conjunto de imagens de cédulas de dinheiro em Real, que será melhor descrito na seção Seção 4.1. De forma básica, é apresentado à RNA um conjunto de N imagens para treinamento e a Rede Neural será responsável por ajustar seu conjunto de pesos sinápticos de forma a convergir para um resultado satisfatório ao final do treinamento, o que significa obter uma boa taxa de reconhecimento, ou seja, mais do que 90% de acerto geral na identificação de cédulas, para imagens nunca antes exibidas a ela durante o treinamento.

Na Fig. 4, tem-se a representação de uma RNA com uma camada oculta, porém é possível criar redes com múltiplas camadas ocultas, o que pode resultar em características interessantes para reconhecimento de padrões em imagens. Neste sentido, cada camada oculta de uma RNA se torna responsável por identificar padrões específicos em imagens e estes padrões servem de base para as próximas camadas aprenderem a identificar novos padrões mais complexos.

Existem diversos modelos de RNAs descritos na literatura, alguns deles são: *Multilayer Perceptrons* (De Azevedo et al., 2000), *Radial Basis Function* (Housson, 1995), *Support Vector Machines* (Haykin, 1999), *Convolutional Neural Network* (Haykin, 1999). Este último, as Redes Neurais Convolucionais, foram utilizadas neste trabalho e serão brevemente explicadas na Seção 2.3.1 a seguir.

2.3.1 Redes Neurais Convolucionais

As RNCs foram inspiradas através de experimentos em animais realizados por Hubel e Wiesel em 1962, com a descoberta que alguns neurônios do cérebro correspondentes ao córtex visual de animais reagiam apenas quando expostos às bordas em determinadas orientações, como na vertical, horizontal ou diagonal. Assim, os pesquisadores descobriram que tais neurônios estavam organizados de tal forma que, juntos, eles eram capazes de realizar uma

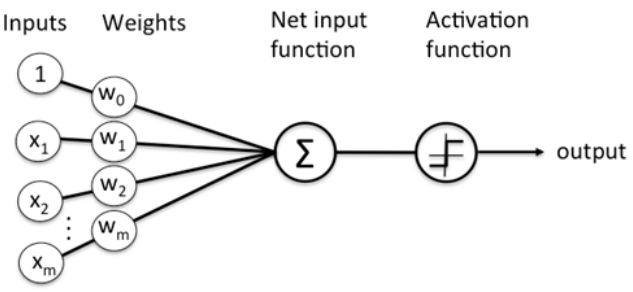

Figura 3: Representação de um neurônio artificial

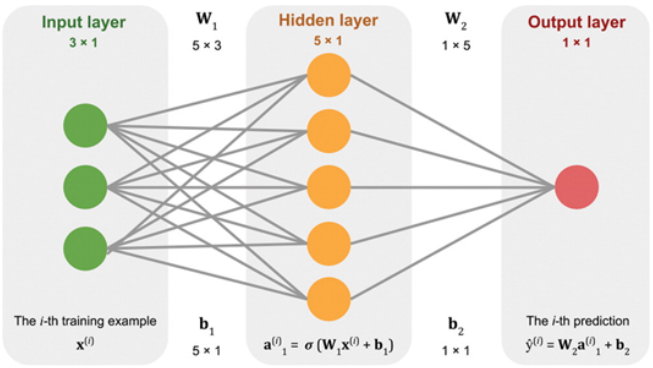

Figura 4: Exemplo de rede neural artificial com uma camada oculta

percepção visual (Deshpande, 2016).

Uma RNC consiste basicamente de uma camada de entrada, camadas de convolução, camadas de pooling e camadas totalmente conectadas, cujos filtros são aplicados a uma imagem dada como entrada para a rede. Segundo Rawat and Zenghui (2017), as camadas de convolução são responsáveis pela extração de características de uma imagem por meio de *kernels* de convolução (filtros).

Estes filtros se deslocam ao longo dos pixels da imagem por meio de deslocamentos (*strides*) a serem feitos a cada operação de convolução, resultando em um mapa de características (*feature map*), ou seja, uma matriz de características extraídas a partir do filtro aplicado.

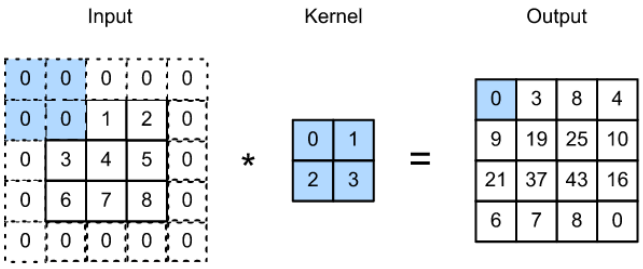

Figura 5: Exemplo de filtro de convolução

A Fig. 5 apresenta um exemplo da aplicação de um filtro de convolução (*kernel*) de tamanho 2x2, conforme destacado em azul no centro da imagem. Este filtro é multiplicado pela matriz de entrada (*input*) de tamanho 5x5, resultando na matriz de características (*output*) de tamanho 4x4. Cada posição da matriz da camada de saída (*output*) é resultado da multiplicação da matriz de *kernel* por posições 2x2 da matriz da camada entrada (*input*).

O deslocamento, ou *stride*, utilizado na Fig. 5 possui tamanho 1 (um), ou seja, os filtros são aplicados deslocando-se pela matriz de 1 em 1 coluna, da esquerda para a direita e iniciando do topo, o que resulta na nova matriz (*output*).

Já as camadas de *pooling* são geralmente utilizadas entre camadas de convolução e são responsáveis por gerarem uma versão menor da matriz resultante da camada de convolução, sendo que existem diversos tipos de camadas

pooling, sendo a mais comum a *max pooling*. A Fig. 6 exibe um exemplo da camada *max pooling* em uma dada matriz de entrada de tamanho 4x4. O tamanho do *max pooling* é 2x2, conforme destacado na cor roxa, resultando assim em uma matriz 4x4 (*output*).

O *max pooling* obtém o número de maior valor onde é aplicado, neste caso sua aplicação é destacada na cor roxo na matriz input, obtendo assim o valor máximo 8, o qual é o primeiro valor (topo esquerdo) da matriz resultante (*output*), este processo é, então, repetido utilizando o deslocamento. O deslocamento ou *stride*, utilizado na Fig. 6, possui tamanho 2 (dois), ou seja, os filtros são aplicados deslocando-se pela matriz de 2 em 2 colunas, da esquerda para a direita e iniciando no topo, o que resulta na nova matriz (*output*).

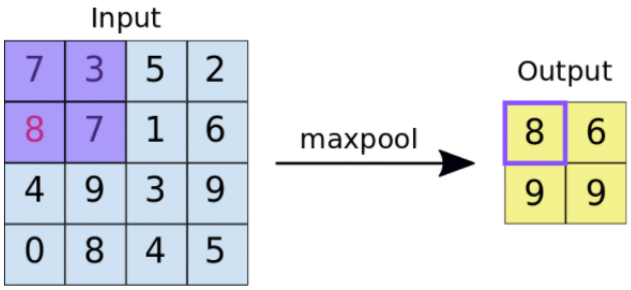

Figura 6: Exemplo de *max pooling*

Por fim, há as camadas totalmente conectadas, as quais possuem o papel de realizar a classificação baseada nas características extraídas pelas convoluções. Geralmente a última camada totalmente conectada utiliza uma função de ativação chamada *Softmax*, a qual resulta em valores entre 0 e 1 para cada classe que o modelo tenta prever (GOOGLE, 2020a). A Fig. 7 apresenta uma visão completa de uma RNC que possui duas camadas de convolução, seguidas de camadas max pooling e, por fim, duas camadas totalmente conectadas.

Após a realização de uma convolução, a qual ocorre nos blocos em vermelho na Fig. 6, também é aplicada uma função chamada ReLu (*Rectified Linear Unit* – Unidade Linear Retificada). Esta função é linear e responsável por ativar o valor de neurônios, de forma a tornar zero valores que estejam menores que zero. De acordo com Data Science Academy (2019) "ReLu é a função de ativação mais amplamente utilizada ao projetar redes neurais atualmente".

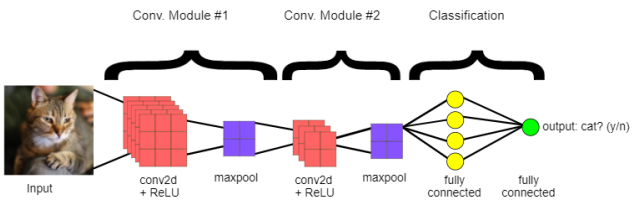

Figura 7: Exemplo completo de uma RNC

3 Trabalhos Relacionados

O trabalho de Doush and AL-Btoush (2017), que leva o título de "*Currency recognition using a smartphone: Comparision between color SIFT and gray scale SIFT algorithms*", teve como objetivo principal desenvolver um aplicativo para *smartphones* que reconhecesse cédulas e moedas do País da Jordânia. Para que tal objetivo fosse alcançado, foram testados e comparados dois algoritmos de visão computacional: color SIFT (scale-invariant feature transform) e gray scale SIFT. Os autores utilizaram um conjunto de 100 imagens para treinamento e 400 imagens para testes, as imagens são referentes a 5 tipos de cédulas e 5 tipos de moedas, com diferentes ângulos, fundos, iluminações e distâncias. Os resultados obtidos com este trabalho, através de testes em 400 imagens, mostraram que: o percentual médio de reconhecimento utilizando o algoritmo color SIFT foi de 71% para cédulas e 25% para moedas, e o tempo médio para reconhecer foi de 72,9 segundos para cédulas e 78,2 segundos para moedas. Utilizando o algoritmo gray scale SIFT, a média de reconhecimento de cédulas foi de 53% e de moedas foi de 20%, além disso o tempo médio foi de 72,4 segundos para cédulas e 80 segundos para moedas.

Outro trabalho, apresentado por Sarfraz (2015), o qual possui o título de "*An intelligent paper currency recognition system*", teve como objetivo desenvolver um sistema de reconhecimento automático de cédulas monetárias do Reino da Arábia Saudita. O sistema proposto é composto por quatro módulos: (i) obtenção da imagem, através de um scanner ou câmera digital; (ii) pré-processamento, incluindo remoção de ruídos da imagem; (iii) extração de características da imagem; (iv) classificação e reconhecimento da imagem. No total foram coletadas 110 imagens, dentre elas: 50 de cédulas limpas, 50 com ruído e 10 inclinadas com ângulo menor que 15 graus. Para a classificação e reconhecimento das imagens, o autor utilizou Rede de Funções de Base Radial (Radial Basis Function Network) que deriva da Função de Base Radial Gaussiana (Gaussian Radial Basis Function) composta de três camadas de neurônios: a camada de entrada, utilizando as características extraídas na fase de préprocessamento; uma camada oculta com 25 neurônios; uma camada final de saída. Este modelo foi treinado e testado com 110 imagens. A média de reconhecimento para imagens normais, sem ruído e sem inclinação foi de 95,37%. Para imagens com ruído e sem inclinação, foi de 91,65%. Para imagens sem ruído e com inclinação menor que 15 graus, foi de 87,5%. Assim, a média de reconhecimento obtido com o trabalho foi de 91,51%, levando em média 3 segundos por imagem para realizar a classificação.

Por fim, o trabalho apresentado por Mombach and Welfer (2013), o qual leva o título de "Proposta de um aplicativo móvel para percepção de imagens estáticas por alunos com deficiência visual", teve por objetivo desenvolver um aplicativo para auxiliar pessoas com deficiência visual na percepção de imagens estáticas no escopo escolar. Para tal, os autores organizaram o trabalho em três etapas: i) observação das técnicas manuais; ii) desenvolvimento do aplicativo de forma iterativa baseado no feedback dos usuários; iii) análise dos testes realizados. Durante a primeira etapa, os autores citam que observaram como os usuários realizavam o reconhecimento de imagens e descobriram

que utilizavam texturas, como palitos ou linhas, para delinear o objeto e assim compreendê-lo. Logo, foi delineado o desenvolvimento de um aplicativo para dispositivos móveis que consiste na leitura tátil de figuras binárias no dispositivo através do toque na tela *touch* e das respostas vibratórias. Os autores destacam que foram feitos quatro ciclos de desenvolvimento e teste com usuários com deficiência visual, sempre ajustando e melhorando aspectos do aplicativo conforme *feedback* dos usuários. Na quarta e última versão do aplicativo, eram emitidos dois sons diferentes quando o pixel fosse preto, o som 1 foi utilizado para pixels de preenchimento da imagem e o som 2 para pixels de borda, facilitando assim o reconhecimento da borda da figura. Os testes finais foram realizados com 2 usuários cegos, 2 usuários com baixa visão e 5 voluntários não cegos que tiveram os olhos vendados. As descrições dadas pelos usuários foram subjetivas em relação às figuras apresentadas e, ainda que não foram as respostas esperadas, elas foram coerentes. Um exemplo são as respostas "formiga" e "duas bolas", para a imagem que representava óculos.

A Tabela 1 apresenta um quadro comparativo das principais características dos trabalhos relacionados descritos. Os critérios definidos para a comparação foram: objetivos, dados utilizados e técnicas utilizadas.

A partir dos estudos realizados, foi possível observar o uso de diferentes técnicas para realizar o reconhecimento de cédulas monetárias (Sarfraz (2015); Doush and AL-Btoush (2017)). O trabalho de Doush and AL-Btoush (2017), baseado em 500 imagens, obteve uma assertividade média no reconhecimento de cédulas de 71%, através do algoritmo *color SIFT* e, através do algoritmo *gray scale SIFT*, de 53%. O tempo médio de processamento obtido foi de 72,9 segundos para o algoritmo *color SIFT* e de 72,4 segundos para o algoritmo *gray scale SIFT*.

Já o trabalho de Sarfraz (2015), baseado em 220 imagens, obteve uma assertividade média no reconhecimento de cédulas de 91,51%, através do método de Rede de Funções de Base Radial, sendo que o processo de reconhecimento leva em torno de 3 segundos para ser realizado. Visando explorar novos métodos para reconhecimento de cédulas, bem como a obtenção de melhor assertividade no processo de reconhecimento das cédulas e diminuição do tempo de resposta, este trabalho utiliza Redes Neurais Artificiais como método de reconhecimento e uma base de dados de 6.500 imagens.

No entanto, o reconhecimento de cédulas por si só não é o todo, é de grande valia ressaltar a importância da inclusão do público-alvo durante o desenvolvimento e validação do sistema, assim como feito em Mombach and Welfer (2013). Sendo assim, este trabalho contou com a participação do público-alvo durante o processo de desenvolvimento e validação do aplicativo.

4 Metodologia

Este trabalho foi concluído a partir da execução das seguintes etapas: (i) Estudos sobre tópicos da Seção 2; (ii) levantamento de trabalhos relacionados da Seção 3; (iii) aquisição de imagens para o *dataset*, descrito na Seção 4.1; (iv) definição da arquitetura de RNA, descrita na Seção 5; (v) ciclos de treinamento e valiação do modelo reconhecedor de cédulas de dinheiro em Real; (vi) ciclos de desenvolvimento e validação do aplicativo para *smartphone*, descrito na Seção 6.

4.1 *Dataset* utilizado

A base de dados deste trabalho é composta de 7 classes de imagens, cada classe se refere ao que a imagem representa. Portanto, são 6 classes representando as cédulas de dinheiro em real, de 2 a 100 reais, e uma sétima classe que representa a não presença de cédulas na imagem.

Sendo assim, a criação do *dataset* se deu basicamente pela captura de fotos de cédulas de dinheiro em Real, por meio de um aplicativo próprio desenvolvido. Este aplicativo possibilitou organizar 6 pastas diferentes para salvar as imagens de cada tipo de cédula separadamente, bem como habilitar o *flash* e a captura em sequência de fotos. A Tabela 2 apresenta a distribuição de imagens por classe.

Durante o processo de captura de fotos das cédulas, foi levado em consideração a ideia de tornar abrangentes os exemplos do conjunto de imagens para cada uma das classes. Portanto, foram fotografadas cédulas em diferentes ambientes, fundos, iluminações, proximidades e também com quantidade similar de fotos de cada lado da cédula, para que o modelo fosse capaz de reconhecer as cédulas em ambos os lados e em diferentes cenários.

Além disso, pensou-se desde o início no público-alvo e em diferentes condições em que poderiam vir a utilizar o aplicativo reconhecedor de cédulas, como em um ambiente escuro. Neste sentido, foram capturadas imagens utilizando o *flash* da câmera, em um ambiente totalmente escuro, buscando representar esta realidade no conjunto de imagens. A Tabela 3 busca dar uma melhor visão desta generalização de imagens para cada classe.

Na Tabela 3, a coluna "Boa e Má iluminação" e suas respectivas quantidades por classe dividem-se praticamente de forma igualitária entre imagens em ambiente bem iluminado e imagens em ambiente com má iluminação. Além disso, o número de imagens frente e verso de cédulas para cada classe e ambiente descrito, está dividido de forma praticamente equivalente, porém suas quantidades não foram contabilizadas com exatidão no processo de criação do *dataset*.

A classe "sem_notas"possui 530 imagens capturadas manualmente e 460 imagens extraídas do *Dataset MIT Indoor Scenes* (Ahmad, 2019), o qual possui 15.620 imagens distribuídas em 67 classes que representam ambientes internos, como cozinha, banheiro, quarto, entre outros. A seleção destas 460 imagens ocorreu de forma manual, onde foram extraídas 30 imagens das 13 seguintes classes: *airport_inside*, *bakery*, *bar*, *bathroom*, *bedroom*, *bookstore*, *bowling*, *buffet*, *cassino*, *children_room*, *classroom*, *closet* e *computerroom*. Em adição foram extraídas manualmente 35 imagens das classes *kitchen* e *meeting_room*, totalizando assim as 460 imagens que, somadas as 530 capturadas manualmente, completam as 990 imagens da classe "sem_notas".

A partir da base de dados geral, a qual contém 6.500 imagens no total (conforme Tabela 2), foram gerados os *datasets* denominados Treinamento e Teste, conforme Tabela 4. A separação destes dois *datasets* foi realizada utilizando 89,23% para Treinamento e 10,77% para Teste.

Tabela 1: Tabela comparativa de trabalhos relacionados

Artigo	Objetivo	Dados utilizados	Técnicas utilizadas
Doush and AL-Btoush (2017)	Desenvolver um aplicativo para smartphones que reconheça moedas e cédulas do país da Jordânia.	Imagens para treinamento (100). Imagens para testes (400). Imagens de cédulas e moedas com diferentes ângulos, fundos, distâncias e iluminações.	*Color SIFT. Gray scale SIFT.*
Mombach and Welfer (2013)	Desenvolver um aplicativo para auxiliar pessoas com deficiência visual na percepção de imagens estáticas no escopo escolar.	Imagens de formas geométricas e linhas. Imagem de peixe, óculos e rato. Todas imagens preto e branco.	Não informado.
Sarfraz (2015)	Desenvolver um sistema de reconhecimento automático de cédulas monetárias do Reino da Arábia Saudita.	Imagens para treinamento (110). Imagens para testes (110).	*Radial Basis Function.*
Este trabalho	Desenvolver um aplicativo para identificação de cédulas de dinheiro em Real que possa ser útil no dia a dia de pessoas com deficiência visual, utilizando como técnica de aprendizado de máquina as Redes Neurais Artificiais.	Imagens de cédulas de dinheiro em Real previamente rotuladas. Imagens para treinamento (4.636). Imagens para validação durante treinamento (1.164). Imagens para teste (700). Imagens de cédulas de ambos os lados, com diferentes ângulos, fundos, distâncias e iluminações (*flash* de câmera incluso).	Algoritmo baseado em Aprendizado de Máquina (Redes Neurais Artificiais Convolucionais).

Tabela 2: Número de imagens por classe

Classe	Nº de imagens
notas_2	921
notas_5	942
notas_10	936
notas_20	946
notas_50	889
notas_100	876
sem_notas	990
Total	6.500

Estes valores foram utilizados a fim de o *dataset* de Teste conter uma quantidade exata de imagens para cada classe, o que resultou na escolha de 100 imagens por classe.

Para o *dataset* de Teste foi realizada uma separação manual, a fim de garantir que o mesmo contenha imagens de diferentes ambientes, abrangendo assim todos aspectos contidos na base de dados como um todo. Dentre as 100 imagens contidas em cada uma das classes no *dataset* de Teste, 35 delas possuem boa iluminação, 35 possuem má iluminação e 30 foram tiradas em ambiente totalmente escuro, utilizando o *flash*. Além disso, metade das imagens de cada tipo de ambiente citado são imagens de frente da cédula e metade de verso, o que torna o *dataset* de Teste bem segmentado, buscando assim ter uma visão concreta do resultado final do modelo.

Já o *dataset* de Treinamento, é subdividido de forma aleatória em dois *datasets* durante o treinamento do modelo classificador: Treino e Validação. Isto é feito em tempo de execução, ficando 80% para o Treino e 20% para Validação, das 5.800 imagens contidas no *dataset* denominado de Treinamento (conforme Tabela 4).

Por fim, a Tabela 5 apresenta a distribuição final dos 3 *datasets* deste trabalho.

5 Reconhecedor de cédulas de dinheiro em Real

Durante pesquisas realizadas, optou-se por utilizar a arquitetura chamada *MobileNet V2* (Sandler et al., 2018). Esta arquitetura foi criada especialmente para ser utilizada em dispositivos móveis, de forma rápida, precisa e com baixo consumo de recursos. A arquitetura consiste basicamente de blocos constituídos pelas camadas internas, conforme ilustra a Fig. 8.

De acordo com Hollemans (2018), a camada de expansão (*Expansion layer*) tem como propósito expandir, des-

Tabela 3: Número de imagens por classe em diferentes ambientes

Classe/Ambiente	Boa e má iluminação	Totalmente escuro (*flash* utilizado)	Total
notas_2	681	240	921
notas_5	723	219	942
notas_10	709	227	936
notas_20	729	217	946
notas_50	698	191	889
notas_100	679	197	876
sem_notas	779	211	990
Total	4.998	1.502	6.500

Tabela 4: Número de imagens dos *datasets* Treinamento e Teste

Classe/Dataset	Treinamento (89,23%)	Teste (10,77%)
notas_2	821	100
notas_5	842	100
notas_10	836	100
notas_20	846	100
notas_50	789	100
notas_100	776	100
sem_notas	890	100
Total	5.800	700

Tabela 5: Número de imagens dos *datasets* finais

Classe/*Dataset*	Treino	Validação	Teste
notas_2	656	165	100
notas_5	673	169	100
notas_10	668	168	100
notas_20	676	170	100
notas_50	631	158	100
notas_100	620	156	100
sem_notas	712	178	100
Total=6.500	4.636	1.164	700

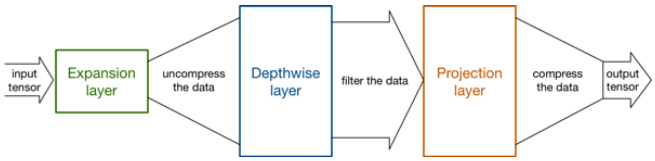

Figura 8: Bloco principal de camadas da arquitetura *MobileNet V2*

comprimir (*uncompress the data*) os dados recebidos (*input tensor*), restaurando-os para sua forma completa. A partir disso, a camada de profundidade (*Depthwise layer*) realiza filtros de convoluções importantes (*filter the data*), dependendo do estágio da rede, sendo ela responsável pela extração de características. Por fim, a camada de projeção (*Projection layer*) compacta os dados novamente para uma dimensão menor (*compress the data*), reduzindo assim a quantidade de informação que trafega pela rede, esta camada também é comumente chamada de *bottleneck layer*.

Tomando como exemplo a Fig. 9, a camada de expansão recebe uma matriz de pixels de 56x56x24, isto é, 56 de altura por 56 de largura e com 24 canais, ou mapa de características da imagem. Exemplo de canais são o RGB (*Red*, *Green* e *Blue*), sendo que cada um dos três canais representa um espectro de cor da imagem.

A camada de expansão então realiza convolução de tamanho 1x1 e expande o número de canais a partir da multiplicação do número de canais da entrada pelo fator de expansão (factor), cujo valor padrão é 6, resultando em uma nova matriz de 56x56x144. Os canais de uma entrada, ou imagem, podem ser vistos como mapas de características desta imagem e são resultados de filtros aplicados a ela.

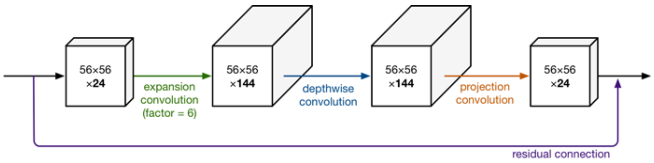

Figura 9: Exemplo de bloco de camadas

A camada de convolução em profundidade, por sua vez, recebe como entrada uma matriz expandida, com diversos mapas de características (canais) para realizar filtros de convoluções de tamanho 3x3. De acordo com Sandler et al. (2018), este tamanho de filtro é o padrão para redes modernas e por este motivo foi adotado na *MobileNet V2*.

As camadas de expansão e convolução em profundidade utilizam normalização em lote (*batch normalization*), seguido da função de ativação ReLu6. De acordo com Brownlee (2019a), *batch normalization* é uma técnica geral que pode ser usada para normalizar dados para uma camada. Já a função de ativação ReLu6 é utilizada para limitar o valor de ativação dos neurônios para um máximo de 6.

Por seguinte, a camada de projeção realiza convolução de tamanho 1x1 e diminui a quantidade de canais e, por consequência, também de informação que trafega pela rede. Esta camada, por sua vez, não utiliza função de ativação ReLu6, pois ela é uma função de característica não-linear e, segundo Sandler et al. (2018), utilizar camadas não-lineares em *bottlenecks* prejudica o desempenho da rede em vários pontos percentuais.

Um outro ponto importante apresentado na Fig. 9 é exibido pela linha roxa, a qual indica uma conexão residual (*residual connection*) entre a entrada e a saída do bloco. Esta conexão de "atalho" entre duas camadas *bottleneck* só existe quando o número de canais é o mesmo entre a entrada e a saída de cada *bottleneck*. De acordo com Sandler et al. (2018), tais conexões residuais possibilitam um treinamento mais rápido e com maior acurácia.

Em suma, a arquitetura *MobileNet V2* original completa é representada na Fig. 10, onde "t" significa taxa de expansão, "c" representa o número de canais de entrada, "n" implica o número de repetições do bloco e "s" o número do passo (*stride*) utilizado na aplicação dos filtros.

A arquitetura contém uma camada inicial de convolução com 32 filtros, seguido de 17 blocos *bottleneck* consecutivamente; posteriormente, seguido de uma camada de convolução 1x1, uma camada de pooling médio global 7x7 (*avgpool*) e, por fim a camada de classificação.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Figura 10: Arquitetura *MobileNet V2*

5.1 Treinamento do modelo

Para a realização do treinamento do modelo, responsável por reconhecer cédulas de dinheiro em Real a partir de imagens, foi utilizada a plataforma *online Google Colaboratory*, a qual dispõe de máquinas que permitem execução de códigos em *Python* em nuvem, bem como disponibiliza acesso gratuito à GPU (*Graphic Processing Unit*), o que é de essencial importância para acelerar o treinamento de modelos de que trabalham com imagens como dado de entrada.

O *framework TensorFlow* foi utilizado durante o treinamento do modelo, que possui ferramentas, bibliotecas e recursos e permite aos desenvolvedores criarem e implantarem aplicativos com tecnologia de *machine learning* (TENSORFLOW, 2020c). O *TensorFlow* utiliza a biblioteca de código aberto *Keras*, a qual fornece abstrações de código essenciais para desenvolver soluções de aprendizado de máquina (KERAS, 2020).

O tamanho definido para entrada de imagens na rede foi 224x224x3, os quais significam, respectivamente: pixels de altura, pixels de largura e número de canais da imagem, neste caso RGB (*Red*, *Green* e *Blue*), que são os canais comuns de cores das imagens. O valor de cada pixel nas imagens RGB varia de 0 a 255, no entanto, a arquitetura *MobileNet V2* define que os valores de entrada sejam normalizados (reescalados) entre -1 e 1.

De acordo com Brownlee (2019b)), valores de entrada sem escala podem resultar em um processo de aprendizado lento ou instável, fazendo com que o processo de aprendizagem falhe. Desta forma, uma fase de pré-processamento ocorre para redimensionar as imagens para 224x224x3 e também para normalizar o valor de seus pixels para entre -1 e 1. A normalização é feita através da função "tf.keras.applications.mobilenet_v2.preprocess_input" (TENSORFLOW, 2020a).

Além disto, com intuito de tornar o reconhecimento de cédulas mais assertivo e genérico, foi utilizada a técnica de aumento de dados (*data augmentation*) através da biblioteca Albumentations (Buslaev et al., 2020). *Data augmentation* tem como objetivo gerar novas imagens a partir de uma imagem base, através de transformações como: rotação da imagem, mudanças de iluminação na imagem, ampliação da imagem, entre outras.

A técnica de transferência de aprendizagem (*transfer learning*) foi utilizada para o treinamento do modelo, que consiste da reutilização de pesos pré-treinados de uma rede em um grande conjunto de dados e tem como intuito diminuir o tempo de treinamento, quando comparado a um treinamento iniciado do zero, bem como o reaproveitamento de caraterísticas de reconhecimento genéricas desta rede pré-treinada, buscando resolver um novo problema (TENSORFLOW, 2020b)). Esta técnica de treinamento é dividida em duas etapas de treinamento: (i) aquecimento da rede (warmup) e ajuste fino da rede (fine tuning).

Para a primeira etapa, comumente chamada de warmup da rede, foram carregados para a rede *MobileNet V2*, os pesos pré-treinados da competição *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC), a qual avalia algoritmos para detecção de objetos e classificação de imagens em larga escala (IMAGENET, 2015). Este *dataset* é composto de 1000 classes diferentes e, portanto, a arquitetura da rede carregada possui em sua camada de saída 1000 neurônios. Assim, fez-se necessário substituir a camada de saída da rede por uma nova camada de saída contendo 7 neurônios, adequando assim a arquitetura da rede pré-treinada para 7 classes, referentes a este trabalho.

Para tanto, foram adicionadas camadas semelhantes às utilizadas em Nguyen (2019), sendo elas: uma camada *GlobalAveragePooling2D*, a qual objetiva diminuir o número de parâmetros de entrada para as camadas densas. Após a camada *GlobalAveragePooling2D*, foram adicionadas duas camadas densas (*Dense Layers*), possuindo respectivamente 1024 e 512 neurônios e possuindo a função de ativação denominada ReLu.

Após estas duas camadas densas, foi adicionada uma camada de *Dropout* com valor de 0,2 (20%), a qual desativa de forma aleatória 20% dos neurônios, visando reduzir o *Overfitting*, que é quando o modelo aprende muito bem a reconhecer os dados apresentados no *dataset* de treinamento, porém não generaliza para outros dados, como o *dataset* de teste, por exemplo.

Por fim, foi adicionada a camada densa de saída contendo 7 neurônios, representando as 7 classes, e utilizando a função de ativação Softmax, a qual resulta em um vetor de probabilidades (de 0 a 1) para cada classe. A Fig. 11 apresenta uma visão geral sobre a arquitetura final da RNC utilizada para o treinamento do modelo.

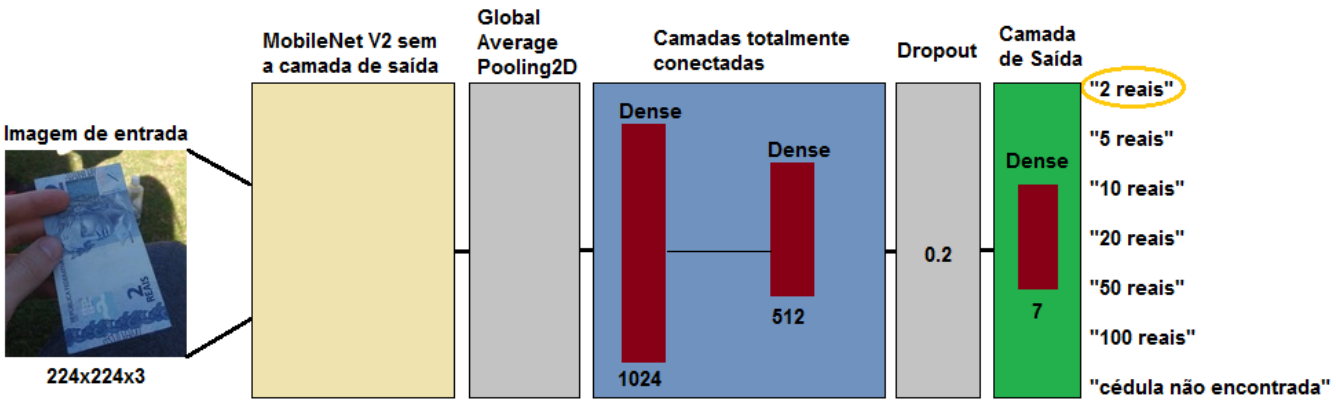

Figura 11: Arquitetura final utilizada no treinamento do modelo

Ainda na etapa de *warmup*, a atualização de pesos da rede durante o treinamento, acontece somente nas camadas finais da rede, enquanto os pesos de todas as camadas da base da rede *MobileNet V2* permanecem congelados, ou seja, não são atualizados durante o treino. Isto é feito para reutilizar o conhecimento armazenado neles, porém agora atualizando somente os pesos das camadas finais, com intuito de resolver um novo problema, o de reconhecer cédulas de dinheiro em Real.

Já durante a segunda etapa de treinamento, comumente chamada de *fine tuning*, foram descongelados os pesos das últimas 54 camadas da rede da arquitetura *MobileNet V2*, a qual contém 154 camadas, desconsiderando a camada de saída, a qual foi substituída por novas camadas, conforme Fig. 11. Durante os treinamentos realizados, foi testado o valor de 34 camadas descongeladas, porém, por fim optou-se por 54 camadas, assim como em (TENSORFLOW, 2020b).

Sendo assim, as 54 camadas da rede base *MobileNet V2* e as 5 camadas finais adicionadas têm seus pesos atualizados durante este segundo treinamento. Com isso, buscou-se realizar um ajuste fino (*fine tuning*) nos pesos de mais camadas da rede, tornando assim as características de reconhecimento de camadas finais da rede mais específicas para o problema em questão, ou seja, reconhecer cédulas de dinheiro em Real.

5.2 Resultados

Após cinco treinamentos completos utilizando diferentes configurações de parâmetros, o resultado final obtido, aplicado sobre o *dataset* de Teste, é ilustrado na Fig. 12, a qual consiste em uma Matriz de Confusão (*Confusion Matrix*) e na Fig. 13, a qual consiste num Relatório de Classificação (*Classification Report*).

Segundo Brownlee (2016), a matriz de confusão é uma técnica para resumir o desempenho de um algoritmo de classificação, fornecendo uma melhor visão sobre quais classes o modelo está acertando e errando.

Na Fig. 12, observa-se a disposição das classes em linhas e colunas, cujas linhas representam as classes verdadeiras (*True label*), enquanto as colunas representam as classes previstas pelo modelo (*Predicted label*). Isto implica que a predição correta para cada classe está no encontro das

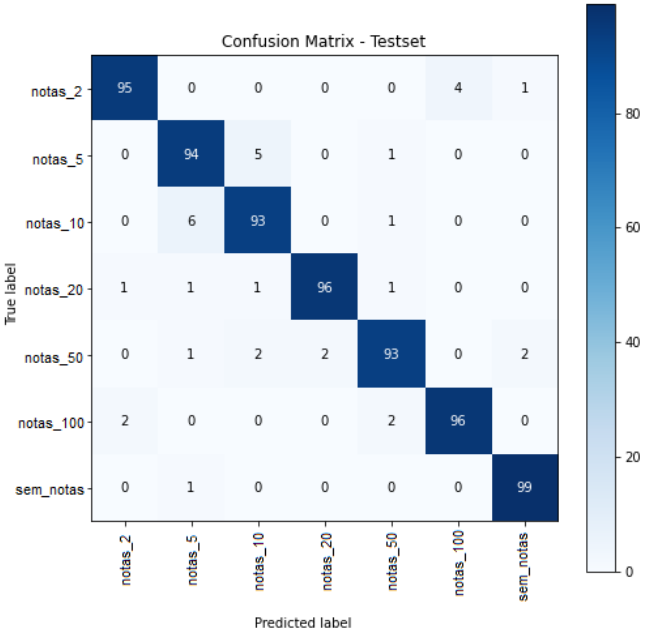

Figura 12: Matriz de Confusão do *dataset* de Teste

linhas e colunas das respectivas classes, isto é, na diagonal principal da matriz. A barra lateral à direita da matriz, indica a variação de valores. Neste caso, os valores variam de 0 a 100, já que o valor máximo de imagens em cada classe é 100, as quais estão presentes no *dataset* de Teste.

A coloração desta barra lateral varia em tom azul, partindo do branco até o azul escuro, o que representa os valores 0 e 100, respectivamente, indicando a quantidade de previsões (*Predicted label*) para uma determinada classe (*True label*), como, por exemplo, o valor 95, na linha 1 (um) e coluna 1 (um), que indica que o modelo previu 95 imagens de notas de 2 reais corretamente. Em contraste, nota-se que o modelo previu 4 imagens de notas de 2 reais como sendo notas de 100 reais, o que pode ser visto no encontro da linha 1 (um) e coluna 6 (seis).

Portanto, quanto menos variação de cor ocorrer na matriz de confusão, ou seja, quanto mais as cores se concentrarem na diagonal principal (previsões corretas), melhor

o resultado obtido pelo modelo. De forma geral, o modelo previu incorretamente 34 de 700 imagens contidas no *dataset* de Teste.

A Fig. 13 apresenta o Relatório de Classificação, também gerado a partir do *dataset* de Teste. Ele apresenta métricas extraídas da previsão do modelo sobre o *dataset* de Teste e, de acordo com Muthukrishnan (2018), é usado para medir a qualidade das previsões de um algoritmo de classificação, quantas previsões são verdadeiras e quantas são falsas.

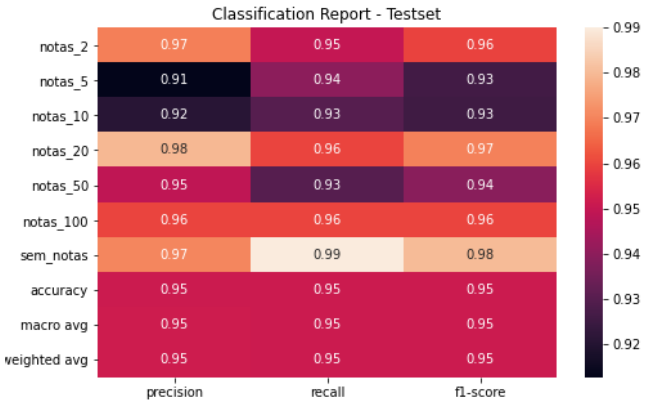

Figura 13: Relatório de Classificação do *dataset* de Teste

O Relatório de Classificação apresenta diferentes métricas, de forma geral e por classe para o modelo. A seguir são detalhadas cada uma das diferentes métricas apresentadas no Relatório de Classificação:

- *Precision*: Quantifica a proporção de previsões corretas, considerando todas as previsões positivas realizadas. Por exemplo, na classe "notas_2" tem-se 0,97 (97%), ou seja, de todas previsões que o modelo indicou ser uma cédula de 2 reais, sendo ela realmente 2 reais ou não, ele acertou 97%. Portanto, *precision* responde a seguinte pergunta: "De todas as previsões feitas para a classe X, quantas realmente estão corretas?"
- *Recall*: Quantifica a proporção de previsões corretas, considerando somente os exemplos que pertencem a esta classe. Por exemplo, na classe "notas_2" tem-se 0,95 (95%), ou seja, de todos os exemplos que realmente são cédulas de 2 reais, o modelo previu corretamente 95%. Neste sentido, *recall* responde a seguinte pergunta: "De todos os exemplos da classe X, qual proporção foi prevista corretamente?"
- *F1-Score*: É um balanceamento entre as métricas precision e recall. Segundo Schade (2019), o cálculo é realizado através da média harmônica, o que gera resultados muito próximos da média comum.
- *Accuracy*: A acurácia representa o percentual de todas as previsões corretas dentre todos os exemplos previstos.
- *Macro avg*: É o cálculo da média aritmética das colunas precision, recall e f1- score sobre todas as classes.
- *Weighted avg*: Possui o mesmo princípio da média aritmética do macro avg, porém considera o peso da quantidade de exemplos por classe para o cálculo.

As diversas métricas descritas tiveram um bom resultado tanto de forma geral, quanto de forma específica entre as 7 classes. Em um aspecto geral, o modelo se mostrou assertivo em 95% dos casos, considerando as 700 imagens contidas no *dataset* de Teste, o qual possui ampla diversidade de imagens conforme descrito na Seção 4.1.

Já de forma específica entre as classes, se mostrou assertivo quase que igualmente, com algumas classes menos assertivas do que outras, como "notas_5" e "notas_10". Uma hipótese para a assertividade destas duas classes ser um pouco mais baixa é a semelhança de cor entre elas. Isto, de certa forma, se justifica na Fig. 12, já que 5 imagens da classe "notas_5" foram previstas como sendo "notas_10" e 6 imagens da classe "notas_10" foram previstas como sendo "notas_5".

Sendo assim, considera-se que o resultado obtido com o treinamento do modelo foi muito bom, possibilitando um reconhecimento assertivo para a grande maioria de imagens de cédulas de dinheiro em Real contidas no *dataset* de Teste.

6 Aplicativo desenvolvido

O aplicativo é denominado "Qual Cédula?" e tem por objetivo auxiliar pessoas com deficiência visual no reconhecimento de cédulas de dinheiro em Real. A primeira versão do aplicativo não possuía a opção de ativar o *flash* da câmera, apenas uma mensagem inicial de "Como utilizar" e a "área da câmera", descrito melhor a seguir. Porém durante testes realizados por um dos usuários, o qual testou o aplicativo em sua primeira versão, indicou que seria interessante ter um modo de habilitar o *flash* para reconhecer imagens capturadas também no escuro.

Então, adicionou-se um botão para habilitar/desabilitar o *flash*, resultando na versão final do aplicativo, o qual pode ser visto na Fig. 14. Ele consiste de uma tela, a qual possui 2 botões no topo e o restante dedica-se à área da câmera, onde o usuário pode dar dois toques para reconhecer a cédula.

Pessoas que possuem deficiência visual e que utilizam *smartphones*, precisam habilitar recursos de acessibilidade para navegar em telas e aplicativos de forma intuitiva. Um destes recursos é o *TalkBack*, o qual de acordo com GOOGLE (2020b) é o leitor de tela do Google incluído em dispositivos *Android*, o qual oferece *feedback* falado para que o usuário possa usar seu dispositivo sem olhar para a tela.

Ao abrir o aplicativo pela primeira vez, será exibida a seguinte mensagem na tela: "Aponte a câmera para a cédula e toque duas vezes para reconhecê-la. Para ativar ou desativar o *flash*, clique no botão de *flash*.", a qual automaticamente será lida pelo leitor de tela que o usuário tenha ativo em seu dispositivo. Quando se possui um leitor de tela ativo, o toque duplo se torna o padrão para acessar menus e aplicativos, enquanto o toque simples apenas descreve o local que o usuário clicou.

Portanto, ao clicar uma vez sobre qualquer lugar na área da câmera, o aplicativo exibirá a mensagem "Área da Câmera", a qual será lida pelo leitor de tela, situando assim o usuário dos limites da tela. O mesmo acontece ao clicar uma vez sobre algum dos botões, cujo leitor de tela simplesmente irá falar o que está escrito nos botões, isto

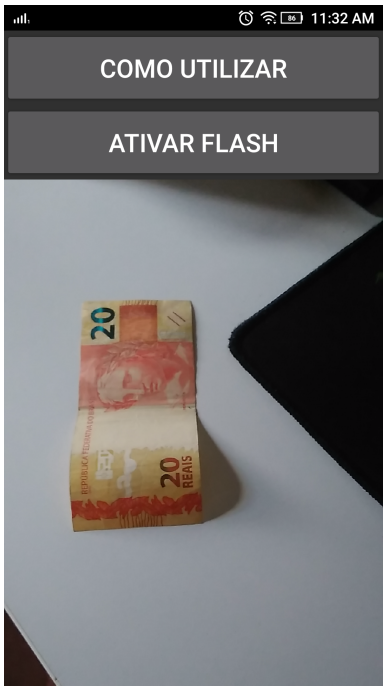

Figura 14: Aplicativo Qual Cédula?

é: "Como utilizar" para o primeiro botão e "Ativar *flash*" ou "Desativar *flash*" para o segundo botão, dependendo do status atual do *flash* da câmera.

Ao dar dois toques no primeiro botão, o leitor de tela irá ler a seguinte mensagem que aparece na tela: "Aponte a câmera para a cédula e toque duas vezes para reconhecê-la.". Já no segundo botão, será exibida em tela e, consequentemente lida pelo leitor de tela, uma das seguintes mensagens: "*Flash* ativado." ou "*Flash* desativado.", situando assim o usuário das ações que executa na aplicação.

O usuário necessita apontar a câmera para uma cédula e dar dois toques na tela para reconhecê-la. Este toque duplo é possível de ser realizado em qualquer lugar da área da câmera, o que facilita a utilização pelo usuário. Ao executar dois toques na área da câmera, a imagem é capturada e classificada pelo modelo.

De acordo com o resultado da classificação, será exibida uma mensagem na tela, a qual novamente será lida pelo leitor de tela do usuário, podendo ela ser: "2 reais", "5 reais", "10 reais", "20 reais", "50 reais", "100 reais" ou "cédula não encontrada". Todo o processo, desde a captura da imagem até a exibição do resultado, acontece de forma *off-line* no próprio dispositivo do usuário e leva em média 417,5ms (*mili* segundos).

6.1 Aspectos de implementação

Para o desenvolvimento do aplicativo foi utilizada a linguagem de programação Java e o desenvolvimento foi voltado para o Sistema Operacional Android, por possuir maior abrangência de mercado no Brasil, com 85,6% em novembro de 2020 (GlobalStats, 2020). A IDE (*Integrated Development Environment*) utilizada para o desenvolvimento foi a Android Studio, por ser um ambiente integrado de desenvolvimento para a plataforma Android, bem como por ser gratuita.

Para possibilitar o uso do modelo em um dispositivo móvel, o modelo TensorFlow resultante do treinamento da rede foi convertido para um modelo TensorFlow Lite. Para isso, foi utilizado o conversor TensorFlow Lite, o qual é uma ferramenta disponível como API (*Application Programming Interface*) Python que converte modelos treinados do TensorFlow em modelos no formato TensorFlow Lite. A partir do modelo convertido, o qual possui 22,6 MB, é possível realizar inferências em imagens capturadas pelo aplicativo. Quando uma imagem é capturada pelo usuário, ela é pré-processada antes de ser usada como entrada para o modelo classificá-la.

A primeira etapa de pré-processamento é o redimensionamento da imagem para 224x224x3, o que significa que possui 224 pixels de altura, 224 pixels de largura e 3 canais de cores, neste caso RGB (*Red*, *Green* e *Blue*). A segunda etapa é a normalização do valor de seus pixels para a faixa entre -1 e 1. Feito isso, a imagem préprocessada é utilizada como entrada na rede, através do modelo TensorFlow Lite alocado na memória do dispositivo móvel, o qual retorna uma lista de 7 predições.

A soma destas 7 predições totaliza 1,0 (100%), pois utiliza a função de ativação Softmax. Portanto, supondo que a imagem de uma cédula de 20 reais foi capturada e utilizada como entrada para a rede, um resultado ilustrativo de valores de predição pode ser visto na Tabela 6.

A classe correspondente ao maior valor de predição retornado pela inferência do modelo é então filtrada e uma decisão é tomada a partir deste resultado: se o valor de predição for maior ou igual a 80%, então julga-se que o resultado é confiável e uma mensagem correspondente à classe resultante é exibida para o leitor de tela. Caso contrário, se o valor de predição for menor que 80%, então é exibido como resultado a mensagem "Cédula não encontrada".

Este valor de 80% foi fundamentado a partir do cálculo da média dos 34 valores de predição incorretos inferidos pelo modelo no *dataset* de Teste. O valor da média ficou em 74,96, o que significa que, em média, o modelo prevê uma classe errada quando o valor de predição está abaixo de 0,7496 (74,96%). Assim, para aumentar a probabilidade do modelo acertar a classe prevista, o limiar escolhido foi 80%.

O tempo de resposta do aplicativo foi medido em um *smartphone* de modelo Lenovo A6020I36, versão 6.0.1 do Android, versão H201 do Hardware, 8 core 1.5GHz de CPU, 2GB de RAM e obteve um valor de 417,5ms (mili segundos) de média, para 10 reconhecimentos realizados.

6.2 Resultados

Para validar o aplicativo desenvolvido foi selecionado um grupo composto por 5 (cinco) pessoas com deficiência visual, as quais foram orientadas quanto ao propósito geral do aplicativo, porém não orientadas quanto à forma de uso do mesmo. Após terem utilizado o aplicativo por alguns dias, foram convidados a responder um questionário de avaliação da aplicação, conforme Tabela 7.

O questionário contém dez questões organizadas no formato da escala de Likert, a qual trata-se de uma das meto-

dologias mais populares e indicadas para realizar pesquisas de opinião (Frankenthal, 2017). Optou-se por utilizar a escala de valores entre 1 a 5 de Likert, os quais referem-se aos níveis definidos no presente trabalho (1 - Muito Ruim; 2 - Ruim; 3 - Razoável; 4 - Bom; 5 - Muito Bom, N/A - Não se aplica).

As colunas P1, P2, P3, P4 e P5 referem-se ao grupo de pessoas com deficiência visual que avaliaram a aplicação, esta avaliação foi feita por ligação telefônica, pois o questionário se mostrou inviável de ser respondido pessoalmente pelo primeiro integrante do grupo a quem o questionário foi enviado.

Além de respostas objetivas, os participantes puderam realizar comentários opcionais quanto à aplicação desenvolvida.

- P1: "Dependendo do ângulo e altura da câmera, às vezes identifica errado o valor da cédula. O aplicativo ficou muito fácil de ser usado. Parabéns pelo teu trabalho e pela iniciativa de querer ajudar pessoas com deficiência visual."
- P2: "Utilização é muito fácil e acessível. Boa sensibilidade da tua parte em criar um aplicativo neste sentido. O aplicativo não se mostrou confiável, há algo para ser melhorado. Nos testes que eu fiz ele confunde muito a nota de 20 com nota de 50 e a nota de 2 com nota de 100."
- P3: "Bem mais eficaz que outro aplicativo que eu usava, o qual lia valores errados e trancava. Funcionou usando o *flash* também. É muito importante ter uma ferramenta que auxilia na identificação. Moro com minha esposa que também é deficiente visual e vou utilizar muito, pois não tenho ninguém para ajudar a reconhecer. Parabéns."
- P5: "Leu bem certinho as notas, errou só uma vez a nota de 50. Acharia mais rápido e acessível fazer a identificação por vídeo. Vou utilizar ele bastante para separar o dinheiro das contas. Parabéns pelo trabalho."

A partir da avaliação das questões organizadas na escala de Likert, foi possível mensurar o nível de satisfação dos usuários em relação ao aplicativo desenvolvido. A satisfação se mostrou boa em maior parte das questões, tendo como resultado da média geral para todas as questões o valor de 4,58 em uma escala de 1 a 5, onde 1 representa "muito ruim" e 5 "muito bom". A Fig. 15 apresenta o gráfico das médias de cada questão.

Dentre as 10 questões, as que obtiveram menor média foram as questões de número 8 e 10, obtendo média 4,2. As questões tratam, respectivamente, sobre a assertividade no processo de reconhecimento e sobre o nível de satisfação de modo geral em relação ao aplicativo.

Acerca da questão 8, sobre assertividade, o usuário P1 destaca que dependendo do ângulo e da altura da câmera, às vezes identifica errado o valor da cédula. Já o usuário P2, o qual avaliou a questão 8 com nota 3, comenta que o aplicativo não se mostrou confiável e que há algo para ser melhorado, pois nos testes que fez ele confunde muito a nota de 20 com nota de 50 e a nota de 2 com nota de 100.

Em contraponto, o usuário P3 destacou que o aplicativo se mostrou mais eficaz que outro aplicativo que ele utilizava, o qual lia valores errados e trancava. Funcionou usando o *flash* também e que vai utilizar muito o aplicativo. Em adição, o usuário P5 comenta que o aplicativo leu bem certinho as notas, errando só uma vez a nota de 50 reais.

Algumas hipóteses para relatos tão diferentes são as seguintes: que o aplicativo funcione com mais precisão em determinados modelos de smartphones; que depende do desempenho/velocidade do dispositivo; que depende da qualidade da câmera do dispositivo. Neste momento, estas considerações ficam para análise como sugestão para trabalhos futuros.

De forma geral, o aplicativo se mostrou positivo em aspectos como: (i) navegação, tendo uma tela simples e intuitiva, não apresentando problemas de navegação para os usuários; (ii) linguagem, apresentando uma linguagem de fácil compreensão nas mensagens de voz; (iii) organização, tendo uma interface com componentes organizados com botões no topo, de modo a facilitar a captura de fotos na região mediana-inferior da tela; (iv) usabilidade, não mostrando dificuldades quanto a sua utilização pelos usuários; (v) acessibilidade, não apresentando impedimentos para pessoas com deficiência visual o utilizarem.

7 Conclusão

O presente artigo apresentou o desenvolvimento de um aplicativo para auxiliar pessoas com deficiência visual no reconhecimento de cédulas de dinheiro em Real, com o uso de Rede Neural Artificial. Em meio a estudos realizados na Seção 2, notou-se o grande número de pessoas com deficiência visual no Brasil, bem como diversas dificuldades enfrentadas por elas, destacando-se para o trabalho a identificação de cédulas de dinheiro. Além disso, estudos realizados sobre Aprendizado de Máquina foram funda-

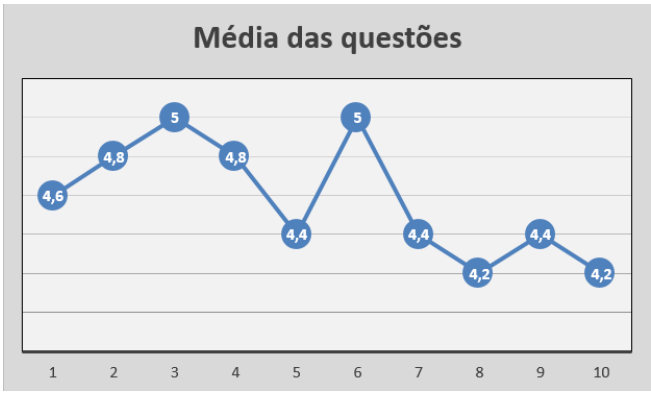

Figura 15: Gráfico da média das questões

Tabela 6: Exemplo de resultado de inferência

Índice da lista	0	1	2	3	4	5	6
Valor de predição	0,01	0,01	0,01	0,93	0,02	0,01	0,01
Classe correspondente	2 reais	5 reais	10 reais	20 reais	50 reais	100 reais	sem notas

Tabela 7: Questionário e respostas dos participantes

Questão	P1	P2	P3	P4	P5	Média Geral
1. Como você classifica a navegação na tela do aplicativo?	4	5	5	5	4	4,6
2. Como você classifica a linguagem utilizada na tela do aplicativo?	5	5	5	5	4	4,8
3. Como você define a organização/disposição das informações na tela do aplicativo?	5	5	5	5	5	5
4. Como você classifica a usabilidade do aplicativo?	4	5	5	5	5	4,8
5. Como você classifica o aplicativo em termos de acessibilidade?	4	5	5	5	3	4,4
6. Como você avalia, como usuário, a utilização de uma ferramenta tecnológica que auxilia no processo de reconhecimento de cédulas de dinheiro em Real?	5	5	5	5	5	5
7. Como você classifica a velocidade do processamento do aplicativo, desde a captura da imagem até o retorno pelo aplicativo?	5	4	5	5	3	4,4
8. Como você classifica a assertividade no processo de reconhecimento de cédula de dinheiro em Real?	4	3	5	5	4	4,2
9. Como você avalia a utilização deste aplicativo como forma de auxílio no reconhecimento de cédulas de dinheiro em Real?	5	2	5	5	5	4,4
10. Qual o nível de satisfação que você avalia este aplicativo de modo geral?	4	3	5	5	4	4,2

mentais para a compreensão e decisão sobre a técnica de RNA utilizada, as Redes Neurais Convolucionais, que estão presentes na arquitetura *MobileNet V2*, tendo esta sido utilizada no trabalho.

Com os resultados demonstrados na Seção 5.2, foi possível verificar que é viável realizar o reconhecimento de cédulas de dinheiro em Real de forma automática por meio de Rede Neural Artificial. A Fig. 13 da Seção 5.2 permite esta conclusão, pois mostra que a arquitetura utilizada, *MobileNet V2*, obteve um percentual médio de 95% de *precision*, *recall* e *f1-score*, considerando todas as classes do *dataset* de Teste. Tendo como pior resultado as classes "notas_5" e "notas_10", obtendo respectivamente, 91% e 92% em *precision* e 94% e 93% em *recall*, resultando assim em 93% na métrica *f1-score*, a qual realiza a média harmônica entre as métricas anteriores.

A participação do público-alvo durante o desenvolvimento e avaliação final do aplicativo desenvolvido foi de grande importância para sua conclusão e resultado, afinal o trabalho foi realizado com o intuito de auxiliá-los. A avaliação final do aplicativo foi realizada a partir de um questionário com 10 perguntas, as quais tiveram suas respostas quantificadas a partir da escala de Likert (1 a 5).

O questionário de avaliação teve como média final o valor de 4,58 e foi realizado através de ligação telefônica feita para cada um dos 5 usuários. Esta forma de contato proporcionou uma conversa aberta com o público-alvo, o qual se mostrou disposto a comentários opcionais, destacados na seção Seção 6.2, os quais serviram de base para diversas conclusões sobre o aplicativo e ideias para trabalhos futuros.

Embora a assertividade do modelo de RNC tenha sido muito bom no *dataset* de Teste, relatos de usuários que testaram o aplicativo, destacados na Seção 6.2, mostram que ainda há algum problema relacionado ao reconhecimento que ocorre no aplicativo. Algumas hipóteses/questionamentos para isso são destacados a seguir, servindo assim de indicação para trabalhos futuros: (i) o aplicativo funciona com mais precisão em determinados modelos de *smartphones*? (ii) uma boa assertividade no reconhecimento depende do desempenho/velocidade do dispositivo? (iii) uma boa assertividade depende da qualidade da câmera do dispositivo?

O presente trabalho possui uma base de dados consideravelmente grande, porém ainda com espaços para melhoria em trabalhos futuros, como: aumento na quantidade de imagens com *flash* e maior diversificação na utilização de métodos de aumento de dados (*data augmentation*) para treinamento do modelo. Em adição, sugere-se para trabalhos futuros a realização do reconhecimento através de frames de vídeo da câmera e não através da captura de fotos, como é atualmente feito. Isto traria agilidade no uso do aplicativo, conforme também destacado pelo usuário P5 na seção Seção 6.2, bem como um aumento na confiança do resultado, uma vez que isto abre a possibilidade de se retornar como resultado a classe mais prevista nos últimos N *frames* de vídeo, por exemplo.

Em suma, considera-se que o objetivo principal do trabalho foi alcançado: desenvolver um aplicativo para identificação de cédulas de dinheiro em Real que possa ser útil no dia a dia de pessoas com deficiência visual, utilizando técnica de aprendizado de máquina, como as Redes Neurais Artificiais. Dos cinco usuários que utilizaram e avaliaram o aplicativo, três deles destacaram que iriam continuar utilizando em seu dia a dia como forma de auxílio na separação de cédulas de dinheiro, enquanto os outros dois, destacaram que de modo geral o aplicativo deve melhorar na questão da assertividade no reconhecimento das cédulas.

Referências

Ahmad, M. (2019). *MIT Indoor Scenes*. Disponível em https://www.kaggle.com/itsahmad/indoor-scenes-cvpr-2019/version/1.

BRASIL (2015). *LEI Nº 13.146, de 6 de julho de 2015. Institui a Lei Brasileira de Inclusão da Pessoa com Deficiência (Estatuto da Pessoa com Deficiência)*. Disponível em http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2015/lei/l13146.htm.

Brownlee, J. (2016). *What is a Confusion Matrix in Machine Learning*. Disponível em https://machinelearningmastery.com/confusion-matrix-machine-learning/.

Brownlee, J. (2019a). *How to use Data Scaling Improve Deep Learning Model Stability and Performanc*. Disponível em https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/.

Brownlee, J. (2019b). *How to use Data Scaling Improve Deep Learning Model Stability and Performance*. Disponível em https://machinelearningmastery.com/how-to-improve-neural-network-stability-and-modeling-performance-with-data-scaling/.

Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M. and Kalinin, A. A. (2020). Albumentations: Fast and flexible image augmentations, *Information* **11**: 125. http://dx.doi.org/10.3390/info11020125.

CONDE (2017). *Definição de cegueira e baixa visão*. Disponível em http://www.ibc.gov.br/images/conteudo/AREAS_ESPECIAIS/CEGUEIRA_E_BAIXA_VISAO/ARTIGOS/Def-de-cegueira-e-baixa-viso.pdf.

Data Science Academy (2019). *Deep Learning Book*. Disponível em https://www.deeplearningbook.com.br/funcao-de-ativacao/.

De Azevedo, F. M., Brasil, L. M. and De Oliveira, R. C. L. (eds) (2000). *Redes neurais com aplicações em controle e em sistemas especialistas*, Visual Books.

Deshpande, A. (2016). *A Beginner's Guide To Understanding Convolutional Neural Networks*. Disponível em https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/.

Doush, I. A. and AL-Btoush, S. (2017). Currency recognition using a smartphone: Comparison between color sift and gray scale sift algorithms, *Journal of King Saud University - Computer and Information Sciences* **29**: 484–492. https://doi.org/10.1016/j.jksuci.2016.06.003.

Frankenthal, R. (2017). *MINDMINERS – Entenda a escala Likert e como aplicá-la em sua pesquisa*. Disponível em https://mindminers.com/blog/entenda-o-que-e-escala-likert/.

Ganssle, G. (ed.) (2018). *Neural networks*, The Leading Edge.

GEEKSFORGEEKS (2019). *Effect of Bias in Neural Network*. Disponível em https://www.geeksforgeeks.org/effect-of-bias-in-neural-network/.

GlobalStats, S. (2020). *Mobile Operating System Market Share Brazi*. Disponível em https://gs.statcounter.com/os-market-share/mobile/brazil.

GOOGLE (2020a). *ML Practicum: Image Classification*. Disponível em https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks.

GOOGLE (2020b). *Primeiros passos no Android com o TalkBack*. Disponível em https://support.google.com/accessibility/android/answer/6283677?hl=pt-BR.

Haykin, S. (ed.) (1999). *Neural networks: a comprehensive foundation*, Upper Saddle River: Prentice Hall.

Hollemans, M. (2018). *Machinethink – MobileNet version 2*. Disponível em https://machinethink.net/blog/mobilenet-v2.

Housson, M. H. (ed.) (1995). *Fundamentals of artificial neural networks*, Cambridge: The MIT.

IBGE (2018). *Censo Demográfico 2010. Nota técnica 01/2018. Releitura dos dados de pessoas com deficiência no Censo Demográfico 2010 à luz das recomendações do Grupo de Washington. 2018*. Disponível em https://ftp.ibge.gov.br/Censos/Censo_Demografico_2010/metodologia/notas_tecnicas/nota_tecnica_2018_01_censo2010.pdf.

IMAGENET (2015). *Large Scale Visual Recognition Challenge (ILSVRC)*. Disponível em http://www.image-net.org/challenges/LSVRC/.

KERAS (2020). *About Keras*. Disponível em https://keras.io/about/.

MATHWORKS (2021). *Deep Learning – What is Deep Learning?* Disponível em https://www.mathworks.com/discovery/deep-learning.html.

Mombach, J. G. and Welfer, D. (2013). Proposta de um aplicativo móvel para percepção de imagens estáticas por alunos com deficiência visual, *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)* **24**. http://dx.doi.org/10.5753/cbie.sbie.2013.487.

Nguyen, H. (2019). *Transfer-learning-with-keras*. Disponível em https://github.com/habom2310/Transfer-learning-with-keras.

Nielsen, M. (2019). *Neural Networks and Deep Learning*. Disponível em http://neuralnetworksanddeeplearning.com/.

Oppermann, A. (2019). *What is Deep Learning and How does it work?* Disponível em https://towardsdatascience.com/what-is-deep-learning-and-how-does-it-work-2ce44bb692ac.

Pila, A. D. (2001). *Seleção de atributos relevantes para aprendizado de máquina utilizando a abordagem de rough sets*, Master's thesis, Universidade de São Paulo. Disponível em https://www.teses.usp.br/teses/disponiveis/55/55134/tde-13022002-153921/pt-br.php.

Rawat, W. and Zenghui, W. (2017). Deep convolutional neural networks for image classification: A comprehensive review, *MIT Press Direct* **9**: 2352–2449. https://doi.org/10.1162/neco_a_00990.

Rezende, S. O. (ed.) (2003). *Sistemas Inteligentes – Fundamentos e Aplicações*, Manole, São Paulo, SP, Brasil.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks, *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. http://dx.doi.org/10.1109/CVPR.2018.00474.

Sarfraz, M. (2015). An intelligent paper currency recognition system, *Procedia Computer Science* **65**: 538–545. http://dx.doi.org/10.1016/j.procs.2015.09.128.

Schade, G. (2019). *iMasters – Machine Learning: métricas para Modelos de Classificação*. Disponível em https://imasters.com.br/desenvolvimento/machine-learning-metricas-para-modelos-de-classificacao.

TENSORFLOW (2020a). *TensorFlow Core v2.4.0*. Disponível em https://www.tensorflow.org/api_docs/python/tf/keras/applications/mobilenet_v2/preprocess_i.

TENSORFLOW (2020b). *Transfira aprendizagem e ajuste fino*. Disponível em https://www.tensorflow.org/tutorials/images/transfer_learning.

TENSORFLOW (2020c). *Uma plataforma completa de código aberto para machine learning*. Disponível em https://www.tensorflow.org/.