# Modelo de Visão Computacional para Auxílio a Deficientes Visuais na Locomoção Urbana

**Ângelo Eduardo Ghellar Tonon**[1], **Carine Geltrudes Webber**[2]

[1]Bacharelado em Ciência da Computação
Área do Conhecimento de Ciências Exatas e Engenharias
Universidade de Caxias do Sul (UCS) – Caxias do Sul – RS – Brasil

[2]D.R. Área do Conhecimento de Ciências Exatas e Engenharias
Universidade de Caxias do Sul (UCS) – Caxias do Sul – RS – Brasil

aegtonon@ucs.br, cgwebber@ucs.br

***Abstract.*** *The walking in street act in a big city can be extremely normal for a citizen. However, many people, that for natural or accidental reasons, have no more a vision capacity. In these cases, something that would be simple become a very challenging mission. In order to assist this task it is proposed a computer vision model. It was trained with obstacle images and urban scenery, for that it can be useful by visually impaired people in your everyday locomotion. For the model development, from systematic revision process, techniques like convolutional neural networks were identified as the most used in similar problems. The model was conceived from seven pre-defined cyclic steps and in the end an accuracy of 83.69% was obtained.*

***Resumo.*** *O ato de caminhar na rua em uma grande cidade pode parecer algo extremamente normal para um cidadão. Entretanto há algumas pessoas, que por motivos naturais ou acidentais, não possuem mais a capacidade da visão. Nesses casos, algo que deveria ser simples torna-se uma missão muito desafiadora.A fim de auxiliar nesta tarefa propõe-se desenvolver um modelo de visão computacional, treinado com imagens de obstáculos e cenários urbanos, para que possa ser útil por deficientes visuais na locomoção do seu dia-a-dia. Para o desenvolvimento do modelo, a partir do processo de revisão sistemática, identificou-se técnicas como as redes neurais convolucionais como sendo as mais utilizadas em problemas similares. O modelo foi concebido a partir de 7 etapas cíclicas pré-definidas e obteve-se ao final uma acurácia de 83,69%.*

## 1. Introdução

Este trabalho se insere no contexto da Visão Computacional, área da Inteligência Artificial (IA), que se ocupa do desenvolvimento de métodos e técnicas de reconhecimento e classificação de imagens [Carvalho 2011, MITCHELL 1997]. Destaca-se os métodos de Redes Neurais *Deep Learning*, que compreendem um conjunto de implementações tais como as Redes Neurais Convolucionais [Academy 2019]. A rede neural para seu funcionamento possui alguns elementos, como o perceptron, que é a menor unidade de processamento em uma rede, sendo que ele possui entradas, realiza um cálculo e retorna um valor [HAYKIN 2003]. Cada camada da rede é composta por vários perceptrons. Eles

possuem conexões com os neurônios das próximas camadas. Porém, eles são independentes entre si e cada um processa a informação recebida de uma forma.

Quando fala-se especificamente em Redes Neurais *Deep Learning*, estas possuem múltiplas camadas e são geralmente utilizadas para classificar informações que um humano realiza no cotidiano, como ouvir um som e saber o que significa, ler um texto, compreender, entre outros [HAYKIN 2003, Carvalho 2011]. Indo além no conceito de redes profundas, temos a Rede Neural Convolucional, desenvolvida para classificar imagens. Por um lado, as redes clássicas obtém os pixels de uma imagem e cada qual era enviado para apenas um perceptron, porém muitas vezes a proximidade ou a distância de certos elementos influenciam na determinação dos padrões. Ou seja, se um chapéu está levemente acima da cabeça de uma pessoa pode-se identificar que ele ou está saindo da cabeça ou que que está caindo na cabeça. Já se não conseguirmos perceber essa separação apenas identificamos que a pessoa está utilizando chapéu. A novidade na Rede Convolucional é que o conceito de proximidade é levado em consideração, ou seja, cada *perceptron* é responsável por processar um grupo de *pixels* próximos, como uma matriz de 3x3, 5x5 e 7x7, [RAVINDRA 2017]. Desta forma, observou-se uma grande melhora no desempenho em tarefas de classificação dos modelos classificadores de imagem.

A área da Visão Computacional tem como objetivo realizar as mesmas funções que a visão humana [BROWLEE 2019]. Assim, os algoritmos podem realizar reconhecimento de objetos, movimentos, emoções expressas pela face e outros. Porém, tal como a visão humana, para que uma imagem seja corretamente identificada, vários fatores além da imagem em si são necessários. Cita-se como exemplo os seguintes fatores: o contexto, o horário, o idioma, entre outros. Um detalhe que pode ser destacado em relação à visão humana é que um computador pode possuir diferentes quantidades e qualidades de câmeras, desta forma, poderá enxergar a uma maior distância, no escuro e em 360 graus.

Em contrapartida, a todas as possibilidades de aplicações que a Visão Computacional pode oferecer, existe um empecilho. Para que um modelo consiga ter uma taxa de acerto aceitável é necessário uma base de dados composta por muitas imagens. Isso se deve a complexidade de classificar uma imagem, pois a taxa de erro cresce conforme a quantidade de objetos a serem identificados aumenta. Para diminuir esse fato é necessário acrescentar amostras de dados de cada objeto. Porém, surge um outro problema, pois será necessário executar muitas vezes o algoritmo de treino para se conseguir uma taxa de acerto aceitável, sendo que a cada execução, quanto mais imagens a base tiver, maior será o custo computacional. Lidar com tais desafios, é tarefa do programador.

Para a implementação das Redes Neurais Convolucionais existem algumas plataformas e bibliotecas. Dentre elas, destaca-se as plataformas TensorFlow [1] e Keras [2], desenvolvidas para a linguagem de programação Python[3]. O TensorFlow agrupa implementações variadas de Redes Neurais *Deep learning*. Dentre elas, para as Redes Neurais Convolucionais podem ser implementadas com várias configurações de camadas e filtros de processamento de imagens. Desta forma, neste trabalho pretende-se utilizar a plataforma TensorFlow. Essa plataforma está disponível via pacote de instalação Ana-

[1]TensorFlow. Disponível em: https://www.tensorflow.org/. Acessado em: 21 mar. 2021

[2]Keras. Disponível em: https://keras.io/. Acessado em: 21 mar. 2021

[3]Python. Python Software Foundation. Disponível em: https://www.python.org/. Acessado em: 18 mar. 2021

conda.

Para posteriormente realizar o treinamento do modelo de rede neural será constituído um conjunto de imagens da cidade de Farroupilha-RS. Esse conjunto é denominado de *dataset*. Ainda assim, para expandir o conjunto, as imagens serão complementadas com *datasets* públicos. Após, as imagens geradas serão pré-processadas de acordo com os padrões de treinamento das redes neurais convolucionais.

Tendo em vista a importância do tema e a necessidade de uma solução para o problema, foi realizada uma pesquisa sobre produtos relacionadas ao tema deste trabalho. Assim, identificou-se que no mercado existem algumas aplicações responsáveis pelo auxílio a deficientes visuais, tais como Be My Eyes[4]. Este artefato serve para apoiar a comunicação por vídeo chamadas entre pessoas videntes que queiram ler e descrever imagens à deficientes visuais. Há também o Eye-d [5], que por suas vez realiza a leitura e descrição com o auxílio de métodos da inteligência artificial de imagens e objetos quando o indivíduo sair na rua. Outro aplicativo semelhante é o Aipoly[6], cujo objetivo é identificar e informar ao usuário os mais variados objetos.

A finalidade deste trabalho é desenvolver um modelo de Visão Computacional que possa identificar e classificar elementos do meio urbano. A fim de que, em um outro momento ou por outros autores, este modelo venha a compor um dispositivo para auxiliar deficientes visuais em sua locomoção. Objetiva-se poder identificar objetos e sinais, tais como: semáforos, placas, pessoas, ruas, árvores e obstruções que estejam no caminho. Para a realização do objetivo principal será necessário um aprofundamento teórico no assunto de classificação de imagens. Também, será importante buscar por trabalhos semelhantes, bem como seus resultados obtidos.

Como objetivos específicos tem-se os seguintes:

1. Identificar o estado da arte em Visão Computacional para auxílio a deficientes visuais.
2. Apontar os métodos e técnicas empregados nesta tarefa, observando-se os melhores resultados.
3. Constituir um corpus de imagens para a tarefa proposta.
4. Implementar um modelo computacional de aprendizado profundo para reconhecer imagens do *corpus*.
5. Avaliar os modelos produzidos e selecionar a melhor arquitetura de Visão Computacional desenvolvida.

## 2. Estado da arte

Para ser possível construir um modelo de Inteligência Artificial para classificar imagens do meio urbano alguns conceitos principais são necessários, como por exemplo o que é classificação e como as redes neurais realizam essa tarefa.

### 2.1. Classificação de imagens

Antes de entender os conceitos que envolvem a Inteligência Artificial é necessário compreender classificação. A classificação no aprendizado de máquina envolve algo muito

[4]Be my Eyes. Disponível em: https://www.bemyeyes.com/. Acessado em: 21 mar. 2021
[5]Eye-d. Disponível em: https://eye-d.in/. Acessado em: 21 mar. 2021
[6]Aipoly. Disponível em: https://www.aipoly.com/. Acessado em: 21 mar. 2021

parecido com a forma com que as crianças aprendem a ler em salas de aula. O professor(a) mostra ao aluno um objeto, como um lápis. Após, o mesmo professor escreve as letras L-Á-P-I-S que compõem o nome do objeto. Desta forma o indivíduo realiza tentativas e erros ao longo do aprendizado até o momento de associar a palavra a um objeto por conta própria.

A classificação de algo em Inteligência Artificial é como o exemplo anterior. O modelo classificador precisa receber as informações dos vários exemplos que se deseja classificar. Estes exemplos em Inteligência Artificial são nomeados de amostras. Então, juntamente com as amostras é necessário informar o nome do objeto que se deseja obter a classificação. Esse nome em Inteligência Artificial é conhecido como rótulo. Desta forma, após o treinamento, a Inteligencia Artificial conseguirá ou não classificar as amostras recebidas em seus respectivos rótulos. Logo, quando há a classificação correta é chamado de aprendizado. Entre outras técnicas de classificação, atualmente existem algumas em uso [Turban et al. 2008], são elas:

- Árvores de decisão
- Análise estatística
- Redes neurais
- Classificador Bayesiano
- Algoritmos genéticos
- Teoria dos conjuntos aproximados

Aprofundando no conceito de classificação, mais especificamente na classificação de imagens, que é um dos principais requisitos deste trabalho, percebe-se a presença de uma maior complexidade, pois, há muitas variáveis envolvidas, como por exemplo: o foco, a luminosidade, o ângulo de captura, a distância entre a câmera e os objetos, a qualidade da imagem e entre outros. Assim para melhor entendimento, os classificadores são divididos em *pixel* a *pixel* e por regiões, [SPRING 2006].

Os classificadores *pixel* a *pixel* utilizam apenas a informação de um *pixel* para achar regiões homogêneas, como é o caso das *Deep Learning*.

Já os classificadores por região utilizam determinadas regiões de textitpixels para identificar regiões homogêneas, inicialmente utilizando as bordas nas figuras para unir regiões, como é o caso da *Convolutional Neural Network* (CNN).

Para que um classificador de imagens consiga identificar classes de objetos, ele precisa inicialmente passar por um processo chamado de treinamento. O treinamento pode ser dividido basicamente em dois tipos: supervisionado e não supervisionado. Há também outras formas de treinamento, porém são variações das duas principais e não serão explanadas neste momento.

O treinamento supervisionado consiste na lógica do aluno que precisa de um professor. Ou seja, as regiões das amostras de imagens são previamente classificadas em rótulos conhecidos por um humano. Assim, o modelo receberá como parâmetros: a imagem e o seu rótulo, tendo que aprender como classificar corretamente os mesmos.

Por outro lado, há o treinamento não supervisionado, que é o caso da lógica do aluno que aprende sozinho em casa. Assim, não há uma classificação prévia nas amostras de imagens, ou seja, o modelo terá que realizar, com base nas imagens disponíveis, a

comparação de características presentes nas figuras e identificar padrões semelhantes que possam ser divididos em grupos. Os grupos encontrados são chamados de *clusters* e o algoritmo responsável por esse processo é chamado de algoritmo de *clustering*, como exemplo o algoritmo K-means.

## 2.2. Redes Neurais Artificiais

Em Inteligência Artificial existem inúmeros algoritmos cujo o objetivo é classificar informações em rótulos. Porém, dentre eles há um grupo em especial que tenta imitar o funcionamento do cérebro humano, as chamadas *Artificial Neural Network* (ANN) ou Rede Neural Artifical (RNA) em português.

Assim quando as primeiras redes surgiram e a ideia de poder replicar o funcionamento da mente humana se expandiu, houveram muitos estudos. Entretanto até o ano de 1969 os estudiosos estavam tão entusiasmados com o assunto, dos quais as promessas e desilusões que todas as discussões causavam, que ao perceberem o pouco progresso ocasionou em uma estagnação do assunto até 1981, os chamados 'anos desiludidos'. O fato ocorrido deve-se em grande parte à baixa capacidade de processamento dos computadores da época, os quais tiveram um grande salto de desempenho a partir dos anos 2000. Inclusive, foi o que ocasionou a utilização de redes neurais em diversas aplicações, como: aviação, previsão, diagnóstico de doenças e muitos outros [Academy 2019] (cap. 2).

Atualmente existem diversas variações de redes neurais, no entanto, há uma em especial cujo o foco está na classificação de imagens, estas são as Redes Convolucionais. Por conta deste trabalho focar especificamente na classificação de imagens foi proposto uma maior concentração de estudo nas CNNs.

## 2.3. Redes Convolucionais para a Classificação de Imagens

Uma CNN compreende uma arquitetura *Deep Learning* para classificação de imagens. Elas também estão no grupo de algoritmos que tentam representar e simular o funcionamento da mente humana. No entanto, diferentemente das simples redes neurais multicamadas, que classificam dados em âmbitos gerais, o foco das CNNs está na Visão Computacional.

O surgimento das primeiras ideias sobre Redes Neurais Convolucionais foram ainda no anos de 1970. Entretanto e principalmente por conta da limitação de *hardware*, foi apenas em 1998 que [Lecun et al. 1998] conseguiu criar uma rede capaz de reconhecer dígitos manuscritos. Assim, o nome dado a esse algoritmo foi de LeNET e posteriormente foi substituído para Rede Neural Convolucional.

Antes das ConvNets, para que um modelo de classificação de imagens fosse treinado, cada *pixel* de uma imagem era associado a um neurônio artificial da camada de entrada. No entanto, na maioria das situações a proximidade com que um elemento da imagem tem com outro é extremamente importante. A proximidade dos elementos fazem com que um determinado objeto ou ação faça sentido. Desta forma, as arquiteturas de redes neurais mais antigas não conseguiam relacionar as partes superiores da imagem com as partes inferiores. Ou então a direita e a esquerda. Por conta deste empecilho, essas arquiteturas ultrapassadas acabavam possuindo conexões inflexíveis entre os neurônios.Por outro lado, os neurônios de uma rede convolucional não são responsáveis por apenas um

*pixel* da imagem. Mas, por uma determinada região de *pixels*. Essa região e seu tamanho é denominado de *kernel*.

Inicialmente a Rede Neural Convolucional recebe a imagem como parâmetro de entrada. Porém, os neurônios não são capazes de processar três informações simultâneas, como os três níveis de cores. Assim, para que seja possível manter todas as características pertinentes para a classificação, e também não ser necessário converter a imagem utilizando filtros em nível de pré processamento, os detalhes de cores *Red, Green e Blue* (RGB) são mapeados em uma matriz tridimensional. A Figura 1 representa uma imagem de tamanho 4x4 mapeada e três matrizes(azul, verde e vermelho). As camadas resultantes deste processo são nomeadas de canais.

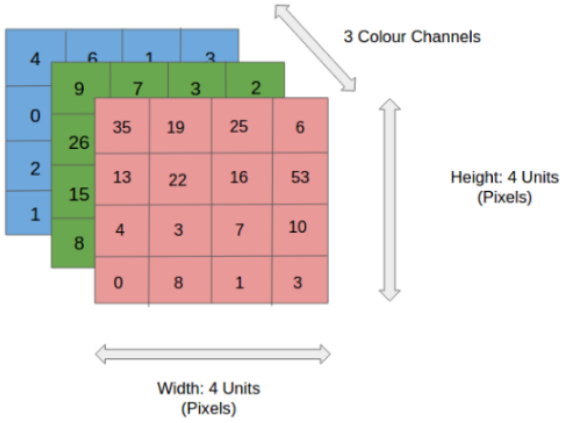


**Figura 1. Representação de imagem convertida em matriz tridimensional**
Obtido de [Academy 2019]. (cap. 40).

Após a entrada da imagem, as informações irão passar por inúmeras camadas, sendo que, essas camadas possuem diferentes funções e finalidades. Assim, são denominadas de:

1. Camada de *pooling*
2. Camada de convolução
3. Camada completamente conectada

A Figura 2 exemplifica da esquerda para a direita o caminho que uma imagem percorre até ser classificada. A imagem do pássaro entra no classificador, passa pelas camadas de convolução, pela camada de *pooling*, pela camada de rede neural completamente conectada e por fim resulta nas probabilidades de cada classe. Assim, para uma melhor compreensão, as partes e camadas que compõem o funcionamento das CNNs foram sub divididas em seções.

### 2.3.1. Campos receptivos

Como dito anteriormente, as redes neurais possuem uma diferença quanto ao mapeamento dos *pixels* das imagens em relação as suas antecessoras. Assim, para compreender como as CNNs conseguem realizar tal tarefa é necessário entender o conceito de campo receptivo. Dessa forma para iniciar, pode ser feito uma demonstração na Figura 3 com a

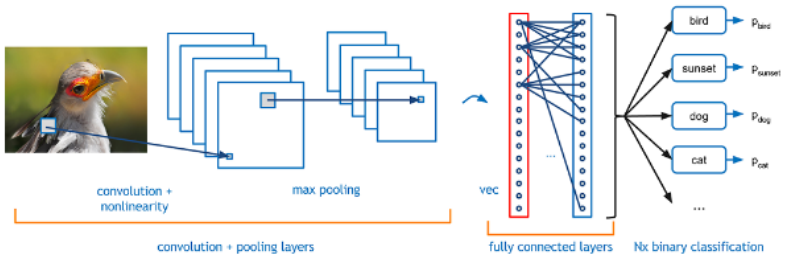


**Figura 2. Exemplo de Rede Neural Convolucional**
Obtido de [Academy 2019]. (cap. 10).

imagem de um semáforo. O campo visual que separa e processa uma determinada parte da imagem de forma individualizada, como no exemplo, é considerado um campo receptivo.

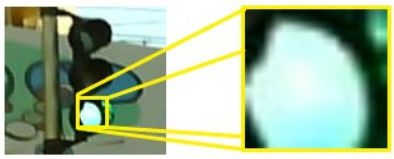

**Figura 3. Exemplo de Campo Receptivo**

Já na rede neural propriamente dita, temos que, cada neurônio da camada oculta será responsável por mapear uma região de neurônios da camada de entrada. Esse mapeamento é realizado pela conexão apenas entre o neurônio da camada oculta com os neurônios que fazem parte da sua região de mapeamento.A camada de entrada por sua vez, poderá armazenar tanto as informações da imagem em sim, quanto conter neurônios que também mapeiam uma camada anterior. Uma demonstração pode ser vista na Figura 4, em que o primeiro neurônio da primeira camada oculta se conecta com um região de 5x5 de neurônios da camada de entrada.

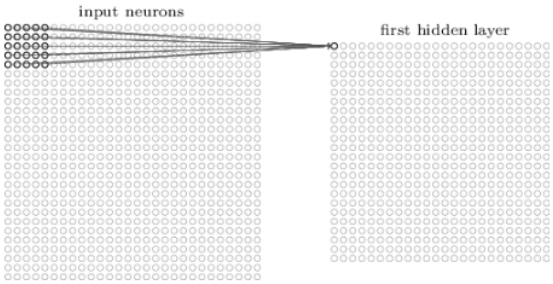


**Figura 4. Campo Receptivo do neurônio 1 da camada oculta**
Obtido de [Academy 2019]. (cap. 41).

Seguindo, todos os neurônios da camada de entrada deverão estar conectados com seus devidos neurônios na camada oculta. Porém um neurônio da camada oculta deverá

estar conectado apenas com os neurônios da sua região de mapeamento. Assim, o tamanho da camada oculta dependerá de três fatores, do tamanho da camada de entrada, da dimensão do campo receptivo(3x3, 5x5, 7x7,...) e principalmente do *stride length*.

O *stride length* é o comprimento de passada. Ou seja, é o valor de deslocamento para o próximo campo receptivo. No caso do *stride length* ser 1 o segundo neurônio terá seu campo receptivo deslocado um neurônio para a direita, como é o caso da Figura 5. O valor do *stride length* e em especial da quantidade de neurônios do campo receptivo podem assumir valores maiores em casos de imagens grandes [Academy 2019].

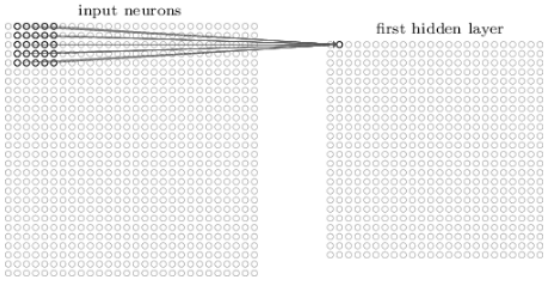


**Figura 5. Campo Receptivo do neurônio 2**
Obtido de [Academy 2019]. (cap. 41).

Então, chega-se na conclusão que o conjunto de todos os neurônios com os seus mapeamentos dos campos receptivos é considerado um mapa de atributos ou características, pois sua principal função é extrair informação útil a partir das imagens originais.

### 2.3.2. Camada de convolução

Os campos receptivos são úteis para encontrar as características em uma imagem. Porém se um olho humano fosse encontrado em uma imagem por uma camada perceptiva, não seria o suficiente para assumir que a imagem é de um rosto humano. Apenas com uma análise completa de todas as características seria possível assumir a presença ou não de um rosto humano, [da Silva et al. 2019] (pág. 192).

Assim, a camada de convolução é constituída por um conjuntos de várias camadas ocultas de características, sendo, que cada uma será responsável por uma característica diferente, podendo então, capturar todas as abstrações dos atributos das imagens em uma informação ainda mais caracterizadora da imagem. Na Figura 6 é possível identificar à esquerda uma camada de 28 x 28 neurônios, os quais poderiam ser os valores de uma imagem. E na direita 3 camadas ocultas, de 24 x24 neurônios cada, de atributos das imagens, formando assim uma camada convolucional.

### 2.3.3. Camada de pooling

As camadas de *pooling* são responsáveis por agrupar as características da imagem obtidas na camada de convolução e geralmente são incluídas após essa, [Academy 2019].

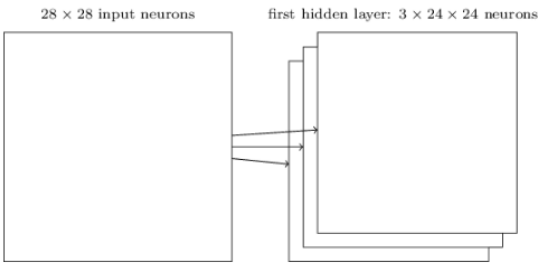


**Figura 6. Mapas de atributos de características**
Obtido de [Academy 2019]. (cap. 42).

O método realiza uma compactação das características através da redução de dimensionalidade, e também a redução de conexões entre os neurônios das camadas de convoluções com as camadas posteriores à camada de *pooling*. O *stride length* e a dimensão da região para a montagem da camada de obtenção de característica também é utilizado para a obtenção da camada de *pooling*. Assim, como pode ser demonstrado na Figura 7, uma região de neurônios 2x2(à esquerda) sendo convertida em um neurônio na camada de *pooling*(à direita).

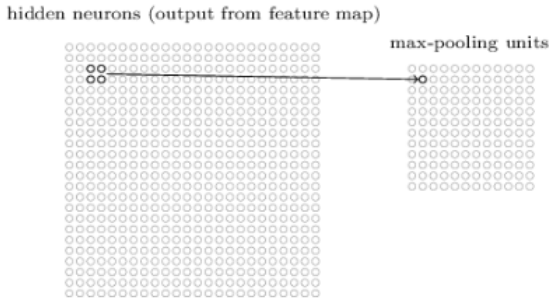


**Figura 7. Exemplificação de um *pooling* de 2x2**
Obtido de [Academy 2019]. (cap. 43).

A obtenção do resultado para o neurônio de *pooling* pode assumir uma de duas formas possíveis, sendo que, com a escolha sendo feita no começo do treinamento, não poderá mais ser alterada. Essas duas formas são na verdade dois cálculos matemáticos diferentes: *max pooling* e *average pooling*.

O *max pooling* ou critério máximo é obtido selecionando o maior valor dos neurônios da região dentro da camada de atributos. Já o *average pooling* ou critério da média é obtido pelo cálculo de média entre todos os neurônios da região, ou seja, a soma de todos os valores dividido pela quantidade total de neurônios, [da Silva et al. 2019] (pág. 205). A Figura 8 demonstra um exemplo dos dois critérios possíveis. À esquerda há uma tabela, que seriam os valores dos neurônios na camada de característica. No canto superior direito há uma tabela exemplificando a camada de *pooling* resultante utilizando o critério máximo. Já no canto inferior direito, também há uma tabela resultante da camada de *pooling*, porém utilizando o critério da média.

No entanto, como visto na seção da camada de convolução, há n possíveis ca-

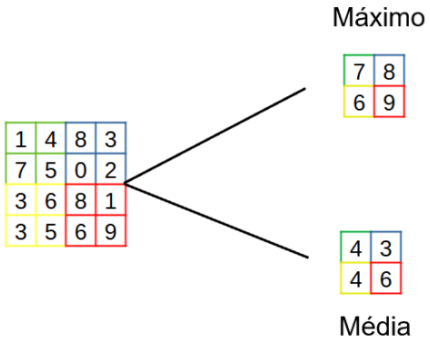


**Figura 8. Cálculo de exemplificação dos métodos de pooling máximo e de média**

madas de atributos. Portanto, a camada de *pooling* também irá conter n camadas, mas com a redução, como é possível visualizar na Figura 9. A camada inicial esquerda com dimensões de 28x28 neurônios. Uma camada convolucional no centro com 3 camadas ocultas de 24x24 neurônios cada. E por fim, na direita, uma camada de *pooling* também de 3 camadas ocultas, porém cada camada com apenas 25% de neurônios em relação a camada anterior.

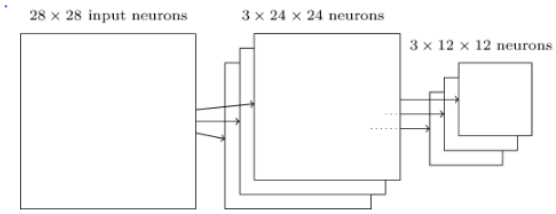


**Figura 9. Esquema de rede neural convolucional com três camadas**
Obtido de [Academy 2019]. (cap. 43).

### 2.3.4. Camada completamente conectada

Por fim, a camada completamente conectada nada mais é, do que uma Rede neural Artificial (RNA) em que todos os neurônios de uma camada estão conectados com todos os neurônios da camada seguinte, ou seja, a camada completamente conectada é uma *Multilayer Perceptron* (MLP).

Desta forma os valores da camada de entrada da MLP serão os neurônios da última camada de *pooling*, enquanto que, a camada de saída da MLP terá a mesma quantidade de neurônios que o conjunto de amostras terá de classes. Porém o resultado não será zero ou um para a classificação das classes, mas sim uma probabilidade da imagem ser as pré-determinadas classes.

## 2.4. Trabalhos Relacionados

Para que fosse possível buscar trabalhos relacionados a este de forma imparcial e analítica foi realizado um processo de revisão sistemática. A revisão sistemática tem como objetivo identificar conhecimento científico previamente desenvolvido por outros pesquisadores.

Para tal fim, levou-se em consideração o guia de [Sampaio and Mancini 2007] no qual foram pré-estabelecidos critérios de coleta e análise de material bibliográfico. Para auxiliar na pesquisa, o portal Science Direct[7] foi definido como a fonte de material científico a ser pesquisado. Após essa escolha, cinco passos foram estabelecidos para a realização desta tarefa:

1. Identificação das palavras chaves
2. Filtragem inicial
3. *Download* do material
4. Análise superficial
5. Análise aprofundada

No passo 1 houve a identificação das palavras chaves. Neste passo foram definidas as palavras chaves da busca como sendo: 'computer', 'vision', 'visually', 'impaired' , 'object' e 'recognition'. Elas foram inseridas de forma combinada no filtro de pesquisa do portal resultando em 2.889 documentos recuperados.

No passo 2 foi realizada a filtragem inicial. Para reduzir de forma significativa a amplitude das informações pesquisadas foi reduzido a documentos de 2019 a 2022 e selecionados apenas dos tipos: 'Review articles', 'Research articles', 'Book chapters' e 'Data articles'. Obteve-se então 363 documentos recuperados.

No Passo 3 ocorreu o *download* do material bibliográfico. Antes da leitura superficial apenas os documentos que disponibilizaram arquivo em formato *Portable Document Format* (PDF) foram baixados. Obteve-se assim 57 documentos para serem analisados.

Em sequência, o passo 4 da análise superficial foi feito. Na análise superficial foi realizada a leitura do título e do resumo dos artigos. Dos 57 documentos, observou-se que apenas 12 tratavam de um assunto relacionado a este trabalho.

Por fim, no passo 5 realizou-se a análise aprofundada. Por meio dela, após a leitura e análise de todos os 12 documentos, foram descartados 7 artigos não relacionados. Sendo assim, apenas 5 artigos que continham tema e objetivo similares foram selecionados e são apresentados nesta seção. Vale ressaltar, que 2 artigos haviam sido selecionados para este trabalho, porém as apresentações dos resultados de um deles eram apresentadas de forma qualitativa, ou seja, com a realização de pesquisa com usuários. Já o outro, possuía métricas de uso de processamento e memória. Assim, a análise comparativa dos resultados deste trabalho com os artigos não seria possível, logo foram removidos.

A Tabela 1 apresenta os 5 artigos resultantes do processo de revisão sistemática e suas métricas de resultado. O primeiro trabalho, desenvolvido por [Sreeraj et al. 2020], trata do uso da visão computacional para auxiliar no dia-a-dia dos deficientes, identificando objetos e obstáculos. O segundo trabalho, desenvolvido por [G S et al. 2020], visa facilitar a navegação de deficiente identificando porta, também como, a direção e a distância da porta. O terceiro trabalho, desenvolvido por [Noceti et al. 2019], trata de um óculos cujo objetivo é auxiliar a comunicação social para múltiplas deficiências. O quarto trabalho, desenvolvido por [Agrahari and Ghosh 2020], trata do reconhecimento de texto com múltiplas orientações em uma cena natural. Por fim, o quinto trabalho, desenvolvido

[7]ScienceDirect. Elsevier B.V. Disponível em: https://www.sciencedirect.com. Acessado em: 18 abr. 2021

por [Elgendy et al. 2021], trata de um sistema de auxílio a localização e navegação para um ambiente interno baseado em marcadores posicionados nas paredes.

**Tabela 1. Medidas de desempenho em estudos relacionados**

| Nro | Referência | Classificação | *Acc* | *Spec* | *Sens* |
|---|---|---|---|---|---|
| 1 | Sreeraj *et al.* (2020) | Faster-RCNN | 91% | 84% | 94% |
| 2 | G S *et al.* (2020) | *CNN - Teste 1* | 97% | - | - |
| | | *CNN - Teste 2* | 84% | - | - |
| | | *CNN - Teste 3* | 92% | - | - |
| | | *CNN - Teste 4* | 87% | - | - |
| | | *CNN - Teste 5* | 92% | - | - |
| | | *CNN - Teste 6* | 98% | - | - |
| | | *CNN - Teste 7* | 97% | - | - |
| | | *CNN - Teste 8* | 84% | - | - |
| | | *CNN - Teste 9* | 92% | - | - |
| 3 | Noceti *et al.* (2019) | Caffe network | 98% | - | - |
| 4 | Agrahari & Ghosh (2020) | *CRF - Horizontal* | 92% | - | 92,1% |
| | | *CRF - Non-Horizontal* | 89,28% | - | 98,2% |
| | | *CRF - Curve* | 88% | - | 87,9% |
| 5 | Elgendy, Sik-Lanyi & Kelemen (2021) | Tiny-YOLOv3 | 99,31% | - | - |

As seções seguintes descrevem as principais características dos trabalhos identificados no processo de revisão sistemática.

### 2.4.1. Auxílio a cegos na identificação de objetos e da distância dos mesmos

O trabalho desenvolvido por [Sreeraj et al. 2020] visa identificar objetos e a distância dos mesmos em uma cena, desta forma, ajuda os deficientes visuais a conseguir identificar obstáculos em seu caminho e a um preço acessível. Para que isso seja possível, o diagrama apresentado na Figura 10 mostra o funcionamento da classificação das imagens. Através da tecnologia *Internet of Things* (IoT), um Arduíno[8] juntamente com um sensor ultrasônico captura a distância e a transforma em som. Já a imagem é capturada por um Raspberry pi[9] acoplado com uma câmera, então, as imagens são enviadas para serem reconhecidas pelo modelo e as regiões que contém os objetos reconhecidos são marcadas com blocos. Por fim o nome dos objetos classificados na cena atual são transformados também em áudio. Já para o treinamento de um modelo de inteligência artificial, os autores utilizaram a biblioteca OpenCV[10] em Python e o algoritmo utilizado foi o de uma CNN modificada, o Fast-RCNN, obtendo os seguintes resultados: 94% de sensitividade, 84% de especificidade, 92% de precisão e 91% de acurácia e apresentados na Figura 11. Por fim, na Figura 12 é possível visualizar a interface do sistema reconhecendo objetos em tempo real. Nesta por sua vez, já é possível visualizar a imagem de uma cena natural com objetos sendo reconhecidos em regiões marcadas em blocos de corres diferentes. As pessoas em rosa, uma garrafa em azul e uma mesa em amarelo.

[8]Arduino. Disponível em: https://www.arduino.cc/. Acessado em: 16 jun. 2021

[9]Raspberry Pi. Disponível em: https://www.raspberrypi.org/. Acessado em: 16 jun. 2021

[10]OpenCV. Disponível em: https://opencv.org/. Acessado em: 16 jun. 2021

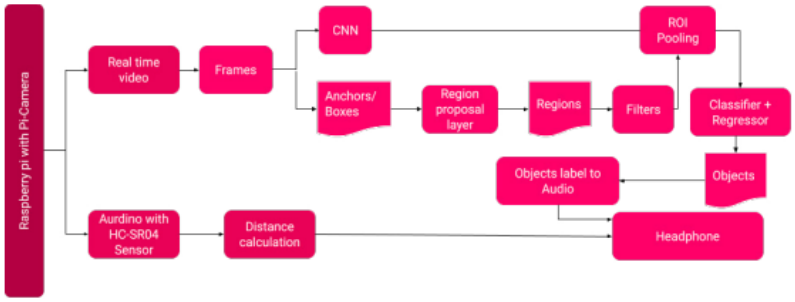


**Figura 10. Diagrama do funcionamento do sistema do trabalho 1**
Obtido de [Sreeraj et al. 2020].

| | | |
|---|---|---|
| Sensitivity(Recall) | 0.94 | TPR=TP/(TP+FN) |
| Specificity | 0.84 | SPC=TN/(FP+TN) |
| Precision | 0.92 | PPV=TP/(TP+FP) |
| Accuracy | 0.91 | ACC=(TP+TN)/(P+N) |

**Figura 11. Resultados de performance do trabalho 1**
Obtido de [Sreeraj et al. 2020].

### 2.4.2. Auxiliar cegos a encontrarem a localização de uma porta através de áudio 3D

O trabalho 2, de [G S et al. 2020], com o objetivo de assistir deficientes visuais em locomoção, foi um estudo de um protótipo de reconhecimento de portas. Assim, para que a pessoa usuária conseguisse identificar o local do destino(a porta), o protótipo informa através de áudio tri-dimensional em um fone de ouvido a distância e a direção. A Figura 13 mostra um pseudocódigo do funcionamento do protótipo, ocorrendo o recebimento de uma imagem, a classificação da imagem e o envio da informação ao sintetizador de voz. Já para a implementação de um modelo, os autores optaram pelo uso da ferramenta SSD MobileNet[11], esta implementa tanto o *Single Shot Detection* (SDD) quanto uma CNN. Após o desenvolvimento do modelo os autores realizaram 9 testes, dos quais 3 estão disponíveis na Figura 14, o melhor resultado de acurácia foi de 97% e o pior foi de 84%. Assim temos que na Figura 15 é apresentado uma porta a esquerda em relação ao usuário sendo reconhecida.

### 2.4.3. Óculos para auxiliar na comunicação social em múltiplas deficiências

O trabalho 3 desenvolvido por [Noceti et al. 2019] é o estudo de um protótipo de óculos para auxiliar pessoas em múltiplas dificuldades. O objetivo é facilitar e melhorar a comunicação social dessas pessoas com o restante da sociedade. Estas funcionalidade estão melhor expressas na Figura 16 que representa o funcionamento do protótipo. As imagens foram obtidas por câmeras de dois celulares de resolução de 1182 x 665, já a câmera do óculos possui uma resolução de 640 x 480 e foram redimensionadas em aproximadamente 60% do tamanho original, cujo objetivo foi melhorar a performance de processamento. Os autores utilizaram o método K-means em conjunto com a extração de forma esparsa dos dados e após realizaram uma normalização nos mesmos para somente

[11]MobileNet. Disponível em: https://keras.io/api/applications/mobilenet/. Acessado em: 16 jun. 2021

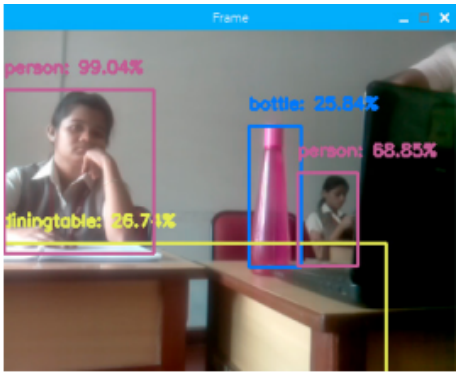


**Figura 12. Resultado em tempo real do reconhecimento de objetos do trabalho 1**
Obtido de [Sreeraj et al. 2020].

```
1. receive input image
2. obtain class and bounding box from model
3. if confidence > threshold and class == 'door':
       send information to 3D Audio Module
   else:
       go to step 1
```

**Figura 13. Pseudocódigo do funcionamento do protótipo do trabalho 2**
Obtido de [G S et al. 2020].

ao final executar o treinamento dos modelos. Diversos modelos foram experimentados, porém a CNN foi o modelo que apresentou o melhor desempenho. Sendo que para a CNN foi utilizado um modelo de 6 camadas, resultando em uma saída de um vetor de tamanho de 4096. Após o processo de treinamento, os autores conseguiram uma acurácia de 98% no melhor modelo, porém não identificaram qual foi o algoritmo utilizado para tal. Outro ponto levantado no trabalho 3 que os autores observaram é o fato de que o modelo baseado em CNN possui de modo geral uma performance de processamento inferior as demais, isso deve-se ao motivo do modelo necessitar mais amostras de imagens para a realização do treinamento, o que ocasionaria uma impossibilidade de uso de forma escalável. Os resultados obtidos no trabalho 3 podem ser visualizados na Figura 17, a qual demonstra as informações de cada categoria de objetos.

### 2.4.4. Detecção de texto em imagens naturais em múltiplas orientações

O trabalho 4 devolvido por [Agrahari and Ghosh 2020], visou abranger o reconhecimento de texto em cenas naturais em múltiplas formas e orientações, como: horizontal, não horizontal e curvos. Assim, a funcionalidade de poder reconhecer texto em placas, nomes de produtos e valores é muito relevante para os deficientes visuais, pelo simples fato, de que a grande parte desses texto não possui um código Braile ou está distante ao toque da pessoa, ocasionando a necessidade de questionar alguém próximo sobre oque está escrito.

As imagens utilizadas para o treinamento dos modelos foram obtidas do IITR *Text*

| | Test Case 7 | Test Case 8 | Test Case 9 |
|---|---|---|---|
| Classification Accuracy | 97 % | 84 % | 92 % |
| Average Depth | 110.31 cm | 136.37 cm | 141.41 cm |
| Top Left Coordinate ($X_1$, $Y_1$) | (26, 17) | (57, 31) | (98, 29) |
| Bottom Right Coordinate ($X_2$, $Y_2$) | (605, 460) | (393, 455) | (358, 454) |
| Normalized Values (X, Y, Z) | (-0.07, -0.03, 3.68) | (-1.48, 0.06, 4.54) | (-1.44, 0.03, 4.71) |

**Figura 14. Casos de teste 1,2 e 3 do trabalho 2**
Obtido de [G S et al. 2020].

**Figura 15. Porta sendo reconhecida no trabalho 2**
Obtido de [G S et al. 2020].

*Datasets*[12], este contém amostras de imagens em orientação horizontal, linear não horizontal e textos curvos, além de conter palavras em inglês, chinês, devanágari e bengali.

A Figura 18 mostra o diagrama de como a imagem obtida passa pela detecção do texto. Inicialmente a imagem é convertida para escalas de cinza, após a imagem passa pela detecção *Maximally Stable External Regions* (MSERS), que é a detecção de regiões com possíveis letras, então as regiões selecionadas são intersectadas com a imagem, após a imagem passa por um algoritmo chamado *Canny Edge Detector*, que é a detecção de bordas, após é realizada uma filtragem de texto utilizando o *Stroke Width Transformation* (SWT) para remover regiões não textuais, e novamente intersectada com a imagem original, por fim é realizado a detecção do texto. A Figura 19 mostra duas imagens de textos sendo reconhecidos, em que a borda em verde é a região que foi detectada a presença de letras.

Por fim o resultados na Figura 20 demonstram que o modelo apresentado pelos autores atingiu 92% de acurácia para textos na horizontal, 89,28% para textos não horizontais e 88% para textos curvos. Assim é possível identificar que textos curvos são mais complexos para métodos classificadores conseguirem reconhecer, pois a acurácia se apresentou um pouco inferior.

[12]IITR Text Dataset. Disponível em: `https://www.researchgate.net/figure/Overview-of-IITR-Text-detection-dataset_tbl1_318670925`. Acessado em: 16 jun. 2021.

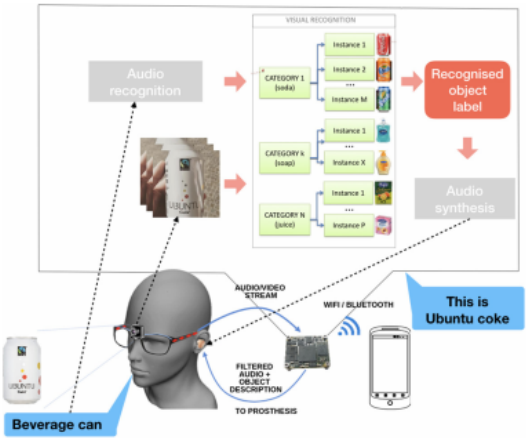


**Figura 16. Representação das funcionalidade do trabalho 3**
Obtido de [Noceti et al. 2019].

**Table 4.1** Recognition rate of the BoF representation on each *Glassense-Vision* dataset use case

| Use case | Background | Rotation | Viewpoint | Occlusion | Avg. RR | Caffe net Avg. RR (%) |
|---|---|---|---|---|---|---|
| Banknotes | 100 | 100 | 97.22 | 89.15 | 97.57 | 75 |
| Cereals | 100 | 100 | 98.19 | 98.95 | 98.92 | 86.16 |
| Cans | 100 | 95.83 | 100 | 100 | 98.21 | 94.64 |
| Water bottle | 96.67 | 100 | 95 | 100 | 97.61 | 90.48 |
| Medicines | 98.14 | 100 | 100 | 100 | 99.49 | 90.91 |
| Deodorant stick | 100 | 100 | 100 | 100 | 100 | 87.88 |
| Tomato sauces | 95.45 | 95.45 | 75 | 93.75 | 92.10 | 93.42 |

**Figura 17. Tabela contendo os resultados do trabalho 3**
Obtido de [Noceti et al. 2019].

### 2.4.5. Sistema de navegação interno para pessoas com deficiência visual baseado em marcadores

O quinto trabalho foi desenvolvido por [Elgendy et al. 2021], o objetivo dos autores foi poder auxiliar deficientes visuais na navegação interna em um centro de compras. Através da utilização de marcadores impressos nas paredes, o indivíduo consegue saber onde está em para qual direção seguir. Na Figura 21 é possível visualizar o esquema de funcionamento do sistema proposto. O usuário primeiro escaneia o marcador e este é enviado para a base de reconhecimento, retornando o local exato em que a pessoa se encontra através de áudio. Em seguida o usuário informa via voz o destino e é calculado com base na posição atual o caminho que deve seguir. As imagens utilizadas para o treinamento foram de marcadores do tipo Aruco, entre outros marcadores, o marcador do tipo Aruco é possível visualizar à direita na Figura 22. Já na Figura 23 é possível visualizar os marcadores impressos nas paredes tanto de forma perpendicular quanto paralelo às paredes, facilitando a obtenção das imagens em diversos ângulos de captura.

Para a classificação das imagens, os autores optaram pela utilização do algoritmo Tiny-YOLOv3 com modificações realizadas pelos próprios autores. O algoritmo Tiny-YOLOv3 é baseado em uma CNN que permite receber uma imagem como entrada, assim, objetivando o reconhecimento em tempo real. Após alguns testes com diferentes versões de modificações do modelo original os autores concluíram que o modelo mo-

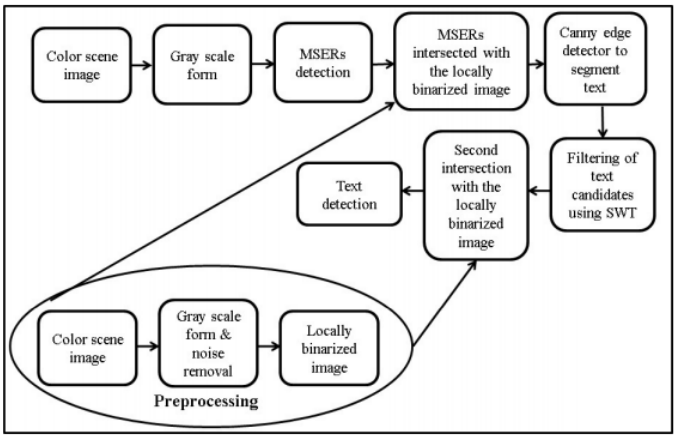


**Figura 18. Diagrama da metodologia proposta do trabalho 4**
Obtido de [Agrahari and Ghosh 2020].


**Figura 19. Texto sendo reconhecido em imagem natural no trabalho 4**
Obtido de [Agrahari and Ghosh 2020].

dificado versão 1 apresentou um melhor desempenho em condições normais do que o modelo original, mostrando 98,27% de acurácia, 100% de recall e 99,13% de F1-Score para a classificação do *dataset* completo, assim podemos visualizar na Figura 24, as 4 versões dos modelos testados nas linhas, e os quatro *datasets* testados nas colunas, com imagens distantes com rotações aplicadas, imagens distantes em condições desfavoráveis, *dataset* contento todas as imagens e *dataset* contento todas as imagens em condições desfavoráveis.

### 2.4.6. Considerações finais obtidas do estudo dos trabalhos relacionados

A partir do estudo realizado, observou-se a incidência do uso das redes neurais em trabalhos relacionados. Por este motivo, considerou-se apropriado o seu uso na proposta de estudo de caso a ser apresentada. Verificou-se também, a enorme presença das Redes Neurais Convolucionais, ou uma adaptação da mesma, na maioria dos documentos, logo, foi optado pela utilização da CNN para treinamento do modelo de reconhecimento de imagens.

Outra questão importante notada foi que os trabalhos possuem alguns objetos específicos para estudo. Um objetiva identificar portas, outro identifica pessoas e objetos em ambiente interno, outro identifica marcadores e outro identifica textos. Assim ao Invés de selecionar uma gama inicial muito grande de possibilidades de objetos, serão selecio-

| Text orientation | Accuracy |
|---|---|
| Horizontal | 92% |
| Non-horizontal | 89.28% |
| Curve | 88% |

**Figura 20. Resultados do melhor modelo do trabalho 4**
Obtido de [Agrahari and Ghosh 2020].

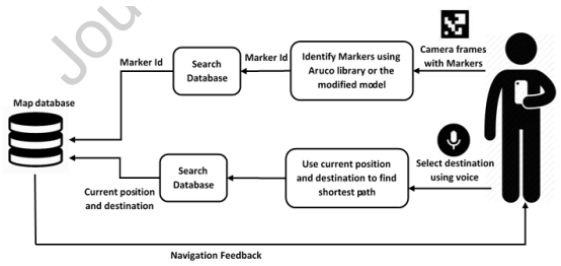


**Figura 21. Esquema de funcionamento do trabalho 5**
Obtido de [Elgendy et al. 2021].

nados apenas alguns objetos do ambiente urbano. Desta forma, será deixado como estudo futuro o reconhecimento de texto em geral e o reconhecimento de objetos de ambientes internos.

Para dar prosseguimento, a seção seguinte descreve o estudo de caso proposto, sua modelagem inicial e os elementos principais.

## 3. Materiais e Método

Os deficientes visuais passam por diversas dificuldades diariamente. Este desafio se torna ainda maior quando a pessoa precisa se deslocar em um ambiente pouco conhecido. Diferente de sua casa, ao ter que se deslocar pelas ruas de uma cidade tudo muda a todo momento. Então mesmo que essa pessoa consiga armazenar na sua mente um caminho e executá-lo diariamente, terá momentos em que ela se encontrará em um local desconhecido, e portanto, precisará de ajuda humana. Assim para tentar minimizar o problema tornando o indivíduo mais independente e seguro, foi proposta uma solução, nas seções abaixo, de um modelo de inteligência artificial que consiga identificar objetos durante o deslocamento do pedestre com deficiência visual.

### 3.1. Estudo de Caso

Para um cego, se deslocar pela cidade é um grande problema. A maioria das cidades não apresentam sinalizações específicas para eles. Embora algumas iniciativas estejam sendo percebidas (sinaleiras com áudio), elas ainda estão disponíveis em poucos locais. Como é o caso na cidade de Belo Horizonte[13] que até 30 de Novembro de 2020 haviam apenas 11 locais com semáforos que utilizam sinais sonoros.

Segundo a Lei Nº 7.853, de 24 de outubro de 1989 a acessibilidade é um direito da pessoa com deficiência visual, tornado um crime para quem desobedecer. Segundo a

[13]Semáforos com avisos sonoros. Disponível em: `https://prefeitura.pbh.gov.br/bhtrans/informacoes/acessibilidade-para-todos/semaforos-com-avisos-sonoros`. Acessado em: 10 jun. 2021.

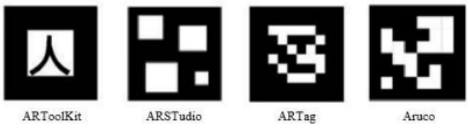


**Figura 22. Exemplos de diferentes marcadores explanados no trabalho 5**
Obtido de [Elgendy et al. 2021].

**Figura 23. Imagem dos marcadores sendo capturados no trabalho 5**
Obtido de [Elgendy et al. 2021].

lei, é obrigatório a utilização de piso tátil seguindo as normas da Associação Brasileira de Normas Técnicas (ABNT) em locais de circulação de pessoas. A Figura 25 ilustra dois tipos de piso tátil. O piso da esquerda representa perigo ou cuidado. Enquanto o piso da direita representa o caminho seguro ou guia.

Já a Lei Nº 11.126, de 27 de Junho de 2005 regulamenta o direito da pessoa com deficiência visual poder adentrar com a presença de cão-guia em ambientes públicos, com a exceção de ambientes de saúde[14].

Há ainda outras tecnologias que visam melhorar a inclusão de pessoas com deficiência visual, tais como: os leitores de tela, as máquinas de braile e os ampliadores de tela. No entanto, ainda há poucos equipamentos que auxiliam no deslocamento em uma calçada. Existem os exemplo do Be My Eyes, Eye-d ou então o Aipoly (aplicações citadas na introdução). Porém, essas aplicações não são focadas nos problemas específicos dos centros urbanos brasileiros. Assim, é neste contexto que se insere este trabalho, propondo o uso de redes neurais para reconhecimento de sinaleiras, obstáculos, buracos, entre outros.

Para contextualizar, apresenta-se alguns exemplos em que obstáculos e má sinalização poderiam atrapalhar o deslocamento de um deficiente visual ou até mesmo ocasionar um acidente. Na Figura 26 é apresentado um caso em que há duas situações. A primeira é a presença de uma grande parada de ônibus que obstrui e dificulta a passagem, tanto para pedestres videntes quanto para pessoas cegas. No entanto, é possível identificar também que não há a presença de piso tátil e é uma calçada irregular, o que poderia facilitar a identificação da obstrução à frente.

Na Figura 27 observa-se um caso que poderia se enquadrar em uma via modelo

[14]Planalto. Disponível em: https://www.gov.br/planalto/pt-br. Acessado em: 10 jun. 2021

| | Far dataset | | | Far Challenging | | | Full dataset | | | Full Challenging | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | P | R | F1 | P | R | F1 |
| **Tiny-YOLOv3** | 95.84 | **100** | 97.88 | 93.29 | 99.99 | 96.52 | 78.32 | 99.98 | 87.84 | 92.52 | **100** | 96.11 |
| **Modified version 1** | **97.21** | 99.83 | **98.50** | 94.43 | 99.90 | 97.09 | 98.27 | **100** | **99.13** | 98.43 | 99.96 | 99.19 |
| **Modified version 2** | 79.88 | 99.63 | 88.67 | 93.94 | 99.56 | 96.66 | 89.17 | 100 | 94.28 | 85.82 | 99.96 | 92.35 |
| **Modified version 3** | 90.86 | 99.29 | 94.89 | **96.85** | **100** | **98.40** | 95.81 | 99.47 | 97.60 | **98.97** | 99.97 | **99.31** |

**Figura 24. Resultados obtidos com precisão(P), Recall(R) e F1-Score(F1) no trabalho 5**

Obtido de [Elgendy et al. 2021].

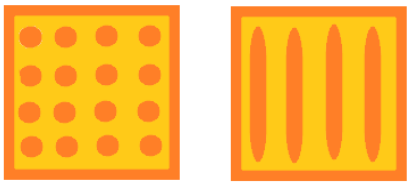

**Figura 25. Ilustração de piso tátil**

para as demais. Contudo, ainda assim há seus perigos. No quadro amarelo, em destaque à direita na mesma figura, é possível identificar um semáforo específico para deficientes visuais. A Associação Farroupilhense de Deficientes Visuais (AFADEV) está localizada no outro lado dessa rua. Na calçada é possível identificar a presença de piso tátil. E no quadro amarelo, logo acima do veículo, é possível identificar um sinal visual-sonoro necessário em saídas de garagens. Mesmo com estas sinalizações, o fato de um veículo estar estacionado sobre a calçada, impede o deficiente visual de prosseguir seu caminho em segurança. Neste cenário reconhece-se a importância e contribuições de um modelo baseado em inteligência artificial.

Na Figura 28 é identificado que a calçada pode ser considerada como regular. Ou seja, não possui buracos, desníveis ou rachaduras. Porém, não há a presença de piso tátil. Também, pode ser visto na imagem tanto uma placa quanto uma árvore. Assim, mesmo que o deficiente possua uma bengala e consiga desviar da placa e do tronco da árvore, ele não conseguirá perceber que os galhos da árvore estão tão baixos que poderão até perfurar um olho de quem estiver passando.

Outro caso de risco encontrado nas calçadas de Farroupilha foi o da Figura 29. Na imagem os quadros em amarelo são antigos canteiros de árvores e arbustos que não estão mais ali. Ou seja, que em algum momento foram removidos e esquecidos. Toda via, essa situação pode representar muitos outros casos de pequenos objetos fixos nas calçadas que poderão ocasionar uma queda. Tanto para deficientes visuais quanto para demais pessoas que possuem algum tipo de dificuldade de locomoção. Também é possível identificar (porém com um pouco de dificuldade por conta da claridade solar)no quadro superior um poste de energia elétrica exatamente no meio da calçada.

Como em qualquer centro urbano há a presença de inúmeras pessoas circulando na mesma via. Assim, o maior desafio para um deficiente é tentar não esbarrar ou até mesmo não derrubar outras pessoas com sua bengala. Na Figura 30 é possível identificar a

**Figura 26. Caso de obstrução por parada de ônibus**

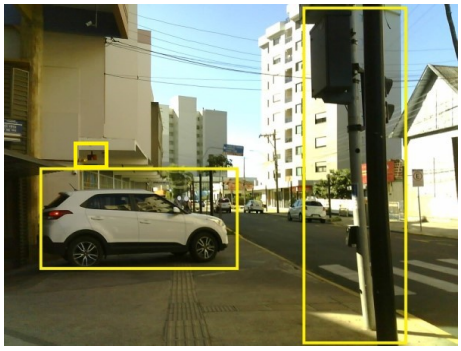

**Figura 27. Caso de obstrução por veículos**

obstrução causada por um árvore e por pessoas. Caso um deficiente esteja passando nesse exato momento, ele poderá fazer com que essas duas senhoras tropecem em sua bengala. Ou então, ele poderá colidir com quem estiver passando ou com a própria árvore.

### 3.2. Percurso Metodológico

Para seguir uma sequência técnica de desenvolvimento do modelo foram organizadas 7 etapas. Estas etapas basearam-se no método de desenvolvimento de uma RNA proposto por [Turban et al. 2008]. Leva-se em consideração que essa seleção prévia de tarefas e etapas é fundamental para o desenvolvimento de qualquer trabalho. Caso ela não seja realizada, problemas como definição incorreta das ferramentas e algoritmos podem surgir em uma etapa muito avançada, ocasionando trabalho extra para corrigir defeitos.

O trabalho foi desenvolvido a partir das seguintes etapas:

1. Seleção de Plataformas, Ferramentas e Algoritmo Utilizados
2. Construção do conjunto de imagens (*dataset*)
3. Pré-processamento das imagens
4. Definição de critérios de avaliação para o modelo
5. Definição da estrutura e aplicação dos algoritmos de Redes Neurais
6. Seleção da melhor configuração de Rede Neural para a tarefa de classificação de imagens
7. Elaboração da justificativa da escolha do melhor modelo

**Figura 28. Caso de obstrução por árvores**

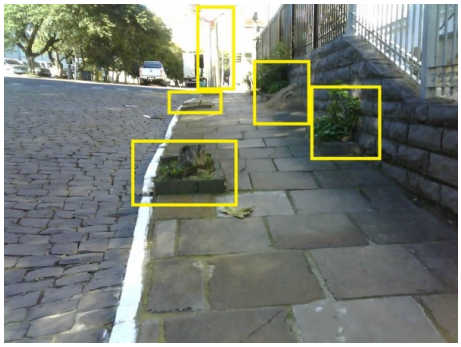

**Figura 29. Caso de obstrução por objetos sobressalentes nas calçadas**

O fluxograma na Figura 31 ilustra essas 7 etapas propostas, bem como os ciclos de retorno à etapas anteriores. Assim, na seção de resultados é apresentado o desenvolvimento realizado em cada etapa.

## 4. Desenvolvimento das Etapas e Resultados

A presente seção descreve o desenvolvimento e resultados obtidos em cada etapa do trabalho.

### 4.1. Etapa 1 - Seleção de plataformas, ferramentas e algoritmos utilizados

O modelo de Inteligência Artificial utilizado foi uma Rede Neural Convolucional. Portanto, algumas ferramentas foram necessárias para facilitar o desenvolvimento de um modelo minimamente viável.

A linguagem de programação utilizada para o desenvolvimento do modelo foi *Python*. Essa escolha levou em consideração a enorme gama de bibliotecas que facilitam o aprendizado de máquina e aprendizagem profunda em *Python*, tal como *Scikit learn*[15], TensorFlow, Keras e entre outras.

[15]Scikit learn. Disponível em: https://scikit-learn.org/stable/. Acessado em: 10 jun. 2021

**Figura 30. Caso de obstrução por outras pessoas**

Utilizou-se as bibliotecas TensorFlow e Keras. O TensorFlow fornece as ferramentas necessárias para construir uma rede neural convolucional. O Keras auxilia no processo de construção, treinamento e avaliação do modelo.

A Figura 32 apresenta os logotipos das ferramentas utilizadas. A linguagem *Python* se encontra acima e maior que as demais(para representar a sua importância). A ferramenta Jupyter Notebook[16] localizada abaixo da linguagem *Python*. Por fim, as bibliotecas Keras e TensorFlow estão nas extremidades.

A Figura 33 mostra uma captura de tela de um trecho de código do tutorial da biblioteca TensorFLow. Esse trecho informa como importar a biblioteca na linguagem *Python*.

## 4.2. Etapa 2 - Construção do conjunto de imagens

As imagens para treinamento e teste do modelo classificador foram obtidas com o auxílio de um protótipo de captura de imagens desenvolvido pelo próprio autor, resultando em aproximadamente mil imagens. O conjunto das imagens obtidas foi complementado com imagens públicas do Open Images Dataset V6[17].

Para ser possível capturar as imagens de forma sistemática foi desenvolvido um protótipo de captura de imagens, cujo, o objetivo deste protótipo é ser utilizado enquanto um pedestre se locomove em um centro urbano.

Para a construção do protótipo foi utilizado uma placa de desenvolvimento chamada ESP32-CAM. Essa placa possui como base o microcontrolador ESP32. Assim, entre outras funcionalidades, a placa possui uma câmera de 2 mega *pixels* e uma entrada para micro-cartões SD. A Figura 34 possui um exemplo deste dispositivo. Na imagem superior(frente) se encontra a câmera e a entrada micro-SD. Enquanto na imagem inferior(verso) está o micro controlador ESP32 acoplado na placa.

No entanto, a placa não possui fonte própria de energia e muito menos um *case* (acessório fundamental para a proteção do dispositivo). Assim, foi necessária a construção desse equipamento. A Figura 35 mostra a parte externa do protótipo já finalizado. Um botão liga/desliga e um *display* de porcentagem de bateria à esquerda e a

[16]Jupyter Notebook. Disponível em: https://jupyter.org/. Acessado em: 16 jun. 2021

[17]Open Images Dataset V6. Disponível em: `https://storage.googleapis.com/openimages/web/index.html`. Acessado em: 15 out. 2021

câmera à direita. Já a Figura 36 mostra a parte interna com a bateria à baixo e a placa ESP32-CAM acoplada em uma placa de prototipagem à cima.

Como qualquer outra placa desse gênero foi imprescindível o desenvolvimento de um código para seu funcionamento. Sendo assim, foi programado para que o dispositivo capturasse uma imagem a cada segundo. Após a captura, a imagem é salva em um micro cartão SD com um código sequencial. Este código sequencial serve para não repetir o nome da imagem e por consequência perder imagens já capturadas. Para esse código sequencial não ser perdido quando o dispositivo for desligado ele é salvo na EEPROM (memória secundária interna no micro controlador) toda vez que uma imagem for corretamente capturada.

### 4.3. Etapa 3 - Pré-processamento de imagens

Na etapa de pré-processamento de imagens foi realizado a classificação e separação das amostras obtidas na etapa 2, bem como, a organização da estrutura de arquivos nos padrões necessários para inserir o conjunto de imagens em um *dataset* da biblioteca *TensorFlow*.

As imagens cruas (obtidas diretamente de câmeras) foram segmentadas com recortes. Ou seja, a partir de uma imagem abrangendo um centro urbano foram recortadas as partes que continham apenas os objetos de estudo. Por conta das grandes cidades serem um ambiente voluptuoso em detalhes, apenas 8 classes de objetos foram selecionados para inserir no treinamento do classificador, elas são as seguintes:

1. Bicicleta
2. Faixa de pedestre
3. Pessoa
4. Piso tátil
5. Planta
6. Semáforo aberto
7. Semáforo fechado
8. Veículo

As imagens após recortadas foram armazenas em um diretório para a construção do *dataset*. Dentro deste *dataset* foram criados oito pastas com os nomes de cada classe. Então, cada uma dessas imagem foi classificada e depositada em sua respectiva pasta. A Figura 37 exibe a quantidade de amostras por classe. Com exceção de piso-tátil e semáforo aberto, todas as demais apresentam 300 amostras, totalizando 2179 amostras.

Para visualizar as imagens que compõe o *dataset* a Figura 38 apresenta um fragmento contendo 9 imagens. Cada imagem nessa figura contém a sua respectiva classe na parte inferior. É possível visualizar 4 imagens de pessoas, 1 imagem de faixa de pedestre, 1 imagem de bicicleta, 1 imagem de veículo e 1 imagem de piso tátil. Estas imagens contém todas a mesma dimensão, mesmo que o recorte original seja maior ou menor, isso porque o modelo precisa receber a camada inicial padronizada em uma quantidade fixa de parâmetros. Assim as imagens foram redimensionadas em 64 x 64 *pixels*.

Por conta da impossibilidade de construção de um *dataset* em grande escala tornou-se necessário realizar um processo chamado de *augmentation*, ou aumento do *dataset*. Esse aumento é realizado por meio da transformação de uma imagem utilizando

alguns parâmetros, tais como: taxa de distorção aleatória, taxa de *zoom* aleatório, taxa de contraste aleatório, taxa de rotação aleatória, espelhamento vertical aleatório, espelhamento horizontal aleatório e outros. No caso deste trabalho foram utilizados os seguintes:

- Espelhamento randômico na horizontal
- 10% de brilho randômico
- 20% de rotação randômica
- 20% de *zoom* randômico
- 1% de contraste randômico

O espelhamento apenas na horizontal faz com que não gere imagens de ponta-cabeça, o que seria pouco provável de acontecer. O brilho auxilia no reconhecimento de imagens de forma independente da luminosidade. A rotação permite que as imagens sejam capturas com a câmera em ângulo e não apenas na horizontal. Enquanto o *zoom* auxilia que imagens de diferentes distâncias sejam bem classificadas. A Figura 39 ilustra um exemplo de uma mesma imagens aplicada 9 vezes o algoritmo de aumento de *dataset*. Para aumentar o *dataset* como um todo foi utilizado uma taxa de aumento igual a 20, isso significa que para cada imagem será obtida a imagem original e mais 19 imagens randômicas do algoritmo de *augmentation*. Considera-se que esse valor elevado poderia gerar imagens repetidas, porém, a grande quantidade de parâmetros aleatórios configurados para o algoritmo torna essas imagens distintas, fornecendo assim uma grande melhoria na taxa de acerto.

Por fim as imagens precisaram passar por uma normalização, que é a transformação dos valores RGB (que variam de 0 a 255) para valores decimais que variam entre 0 e 1. Esse processo é necessário, pois o modelo trabalha apenas com valores nesse intervalo.

### 4.4. Etapa 4 - Definição de critérios de avaliação para o modelo

Nessa etapa foram definidos os critérios para avaliar a qualidade do modelo gerado por determinado treinamento. Foi utilizado inicialmente para fins de testes, uma validação por *Hold Out*. Por conta de que o *Hold Out* é o métodos mais simples para ser desenvolvido. Contudo, posteriormente utilizou-se o método de Validação Cruzada de 10 partições. Segundo autores, o método de Validação Cruzada oferece resultados que permite uma análise mais detalhada do comportamento do modelo [Faceli et al. 2011][MITCHELL 1997]. No intuito de realizar uma predição final utilizando dados que não passaram pelo treinamento e nem pelo teste foi definido que 10% das amostras deveriam constituir um *dataset* exclusivamente para essa função, assim, os 90% restantes foram utilizados para treinamento e testes através do mecanismo de *cross-validation*, totalizando 217 amostras para a predição e 1962 amostras para treino e teste.

### 4.5. Etapa 5 - Definição da estrutura e aplicação dos algoritmos de redes neurais

Nessa etapa a arquitetura inicial da rede foi construída seguindo a documentação da biblioteca TensorFLow. A Figura 40 apresenta como criar uma estrutura de modelo de Rede Convolucional. A instrução 'add' indica a adição de nova camada. Uma camada 'Conv2D' é uma camada convolucional e o 'MaxPooling2D' é um camada de *pooling* utilizando o algoritmo do maior valor. Já a Figura 41 exibe um trecho de código de exemplo para a compilação e treinamento modelo criado.

Tendo como base o modelo inicial, alguns testes de modificações foram realizados. Camadas de convolução e de *pooling* foram criadas e removidas, também como, camadas e valores foram alterados na camada densa. Por fim, foi verificado que as alterações nas camadas de convolução e de *pooling* não demonstraram melhoria no desempenho do classificador(entende-se por desempenho não a velocidade de processamento, mas a capacidade de classificar corretamente um dado de entrada). Por outro lado, o aumento da camada densa melhorou de forma significativa os resultados.

Para a implementação do treinamento foi utilizado o método de Validação Cruzada através da biblioteca Sklearn. O método tem como funcionalidade realizar um *split* do *dataset*. Para exemplificar, dado o valor de kFold=10, o método irá retornar 90% dos índices para serem utilizados para treinamento, e os demais para teste. Desta forma, um laço de repetição foi necessário para realizar o treinamento das 10 partições, sendo que a cada laço, os índices de treino e de teste precisavam ser obtidos com a função *split*. A Figura 42 ilustra a lógica implementada.

## 4.6. Etapa 6 - Seleção da melhor configuração de Rede Neural para a tarefa de classificação de imagens

Essa etapa foi executada em ciclos, seguido o caminho do fluxograma na Figura 31. Assim, para identificar o melhor modelo foi necessário realizar vários testes. Esses testes possuíram diferenças uns dos outros no sentido das seguintes questões: da estrutura do modelo, da preparação dos dados e dos critérios de avaliação.

Ao final do processo cíclico do desenvolvimento deste trabalho foi obtido uma melhor estrutura para o modelo classificador. Essa estrutura, que pode ser visualizada na Figura 43, possui uma camada convolucional de 32 filtros de tamanho 62x62. Em seguida, tem-se uma camada de *pooling* máximo( *stride* 2x2), após uma camada convolucional de 64 filtros. Segue novamente uma camada de *pooling* máximo(*stride* 2x2). Então, segue uma camada convolucional de 64 filtros. Por fim, uma camada *flatten* foi usada para converter as saídas da camada convolucional para as entradas da camada densa. Por fim, tem-se duas camadas densas, uma dimensionada para 2048 saídas e outra para 8, saídas. A última camada produz a classificação dos resultados do modelo.

Quanto ao treinamento, foi estabelecido a utilização de apenas 10 épocas para treinar o modelo. Poucas épocas ocasionaram o efeito de grande taxa de erro, enquanto muitas épocas tornaram o treinamento um processo muito custoso e na maioria dos casos também ocasionou maior taxa de erro. Assim, após sucessivos testes obteve-se 83,69% de acurácia média. As acurácias de cada partição, bem como o desvio padrão da acurácia média podem ser visualizados na Figura 44.

## 4.7. Etapa 7 - Justificativa do melhor modelo

Tendo como objeto validar a utilização do modelo para classificar imagens do meio urbano. Os 10% de amostras do *dataset* foram passados pelo processo de predição. A Figura 45 ilustra os resultados obtidos a partir da predição. Como resultado foi possível identificar que o modelo reconhece com destaque os semáforos, tanto abertos quanto fechados, pois a precisão está acima de 96%. Por outro lado a precisão das imagens contendo plantas permaneceu em apenas 59% de acurácia. Contudo, não considera-se este um resultado ruim, pois o *recall* ficou em 100%, assim todas as imagens contendo árvores foram corretamente classificadas.

Olhando para a taxa de acerto, os dados utilizados para predição apresentaram uma menor acurácia, de 78,34%. No entanto, havia desbalanceamento das classes das imagens, o que pode ter ocasionado tal desempenho. Assim uma matriz de confusão foi gerada para visualizar graficamente o resultado das predições, ilustrada na Figura 46, é possível identificar que muitas bicicletas foram classificadas como pessoas e plantas, muitas pessoas foram classificadas como plantas e veículos e muitos veículos foram classificados como pessoa e planta. Logo, é possível concluir que a acurácia não pode ser melhor por conta de que o modelo teve dificuldade para separar as características dessas classes. Essa dificuldade pode ter sido influenciada pela alta probabilidade de pessoas e bicicletas estarem juntas, o mesmo ocorre para pessoas, veículos e plantas. Ou seja, não há como capturar uma imagem apenas contendo uma característica, pois sempre estarão juntas. Desta forma, pode-se concluir que é um erro aceitável em alguns casos, por exemplo: identificar pessoa em uma bicicleta.

Através de uma análise mais aprofundada nas imagens incorretamente classificadas, foi identificado que na maioria dos casos o modelo identifica uma boa probabilidade para a classe correta, no entanto, por resultar de acordo com a maior probabilidade acaba retornando uma predição incorreta. Como é o caso ilustrado na Figura 47, é possível identificar que a classe correta, veículo em azul, está um pouco abaixo da classe pessoa, que resultou em 28% de probabilidade. Assim, o modelo torna-se ainda mais viável, para construir um produto, utilizando-o juntamente com outros métodos e não apenas a maior probabilidade.

## 5. Conclusões

A inclusão faz parte de toda sociedade que pretende evoluir, nesse pensamento, o modelo construído poderá auxiliar milhares de deficientes visuais com uma locomoção mais segura. Durante o desenvolvimento a etapa de preparação das imagens, a mais importante e também com maior complexidade, apresentou um grande desafio, pois a tarefa de classificação das imagens amplas de centros urbanos em imagens de objetos mais simples mostrou-se um processo muito manual e dependente de mão de obra humana. Mesmo no conjunto inicial de imagens havendo milhares de imagens, o imenso trabalho de classificação resultou em poucas imagens úteis ao *dataset*. Assim, é perceptível que com o tempo e conforme o *dataset* aumentar o modelo poderá se tornar ainda mais preciso. Ainda na primeira etapa destaca-se a dificuldade de encontrar imagens contendo piso-tátil, devendo-se ao fato de que as cidades implementam muito pouco essa lei, e quando implementam, realizam de forma incorreta(que acabam não ajudando esses indivíduos).

Para desenvolvimento futuro, pretende-se utilizar o modelo concebido para o desenvolvimento de um protótipo público e de baixo custo utilizando *smartfones*. Nesse protótipo também poderiam ser incluídos outras classes de objetos, tais como: paradas de ônibus, postes, placas, animais, produtos de mercados, leitura de texto e entre outros. Além da inclusão de novos objetos ao modelo seria necessário também dispor de um dataset com mais amostrar por cada classe, o que o tornaria ainda mais preciso.

## Referências

Academy, D. S. (2019). Deep learning book. `http://deeplearningbook.com.br/redes-neurais-recorrentes/`. Acessado em 21 mar. 2021.

Agrahari, A. and Ghosh, R. (2020). Multi-oriented text detection in natural scene images based on the intersection of mser with the locally binarized image. *Procedia Computer Science*, 171:322–330. Third International Conference on Computing and Network Communications (CoCoNet'19).

BROWLEE, J. (2019). Deep learning for computer vision. image classification, object detection, and face recognition in python. https://machinelearningmastery.com/deep-learning-for-computer-vision/. Acessado em 26 mar. 2021.

Carvalho, A. (2011). *Inteligência Artificial: Uma abordagem de Máquina*. LTC, São Paulo.

da Silva, F. M., Lenz, M. L., Freitas, P. H. C., and dos Santos, S. C. B. (2019). *Inteligência artificial*. SAGAH EDUCAÇÃO S.A, Porto Alegre.

Elgendy, M., Sik-Lanyi, C., and Kelemen, A. (2021). A novel marker detection system for people with visual impairment using the improved tiny-yolov3 model. *Computer Methods and Programs in Biomedicine*, 205:106112.

Faceli, K., Lorena, A. C., Gama, J., and de Carvalho, A. C. P. L. F. (2011). *Inteligencia Artificial : Uma Abordagem de Aprendizagem de Maqunai*. Grupo GEN, Rio de Janeiro.

G S, A. K., Pon, V. N., Rai, S., and Baskar, A. (2020). Vision system with 3d audio feedback to assist navigation for visually impaired. https://www.sciencedirect.com/science/article/pii/S1877050920306815.

HAYKIN, S. (2003). *Redes Neurais: princípios e práticas*. Bookman, 2 edition.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

MITCHELL, T. M. (1997). *Machine Learning*. McGraw-Hill, New York.

Noceti, N., Giuliani, L., Sosa-Garciá, J., Brayda, L., Trucco, A., and Odone, F. (2019). Chapter 4 - designing audio-visual tools to support multisensory disabilities. In Alameda-Pineda, X., Ricci, E., and Sebe, N., editors, *Multimodal Behavior Analysis in the Wild*, Computer Vision and Pattern Recognition, pages 79–102. Academic Press.

RAVINDRA, S. (2017). What is image recognition and why is it used. https://www.kdnuggets.com/2017/08/convolutional-neural-networks-image-recognition.html. Acessado em 26 mar. 2021.

Sampaio, R. F. and Mancini, M. C. (2007). Estudos de revisão sistemática: um guia para síntese criteriosa da evidência científica. https://doi.org/10.1590/S1413-35552007000100013. Acessado em 5 mai. 2021.

SPRING (2006). Sistema de processamento de informações geo-referenciadas. http://www.dpi.inpe.br/spring/portugues/tutorial/classific.html. Acessado em 28 mai. 2021.

Sreeraj, M., Joy, J., Kuriakose, A., M B, B., Babu, A. K., and Kunjumon, M. (2020). Viziyon: Assistive handheld device for visually challenged. https://www.sciencedirect.com/science/article/pii/

S1877050920312618. Third International Conference on Computing and Network Communications (CoCoNet'19). Acessado em 15 out. 2021.

Turban, E., Sharda, R., avid King, and Aronson, J. E. (2008). *Business Intelligence: Um Enfoque Gerencial para a Inteligência do Negócio*. Bookman, 1 edition.

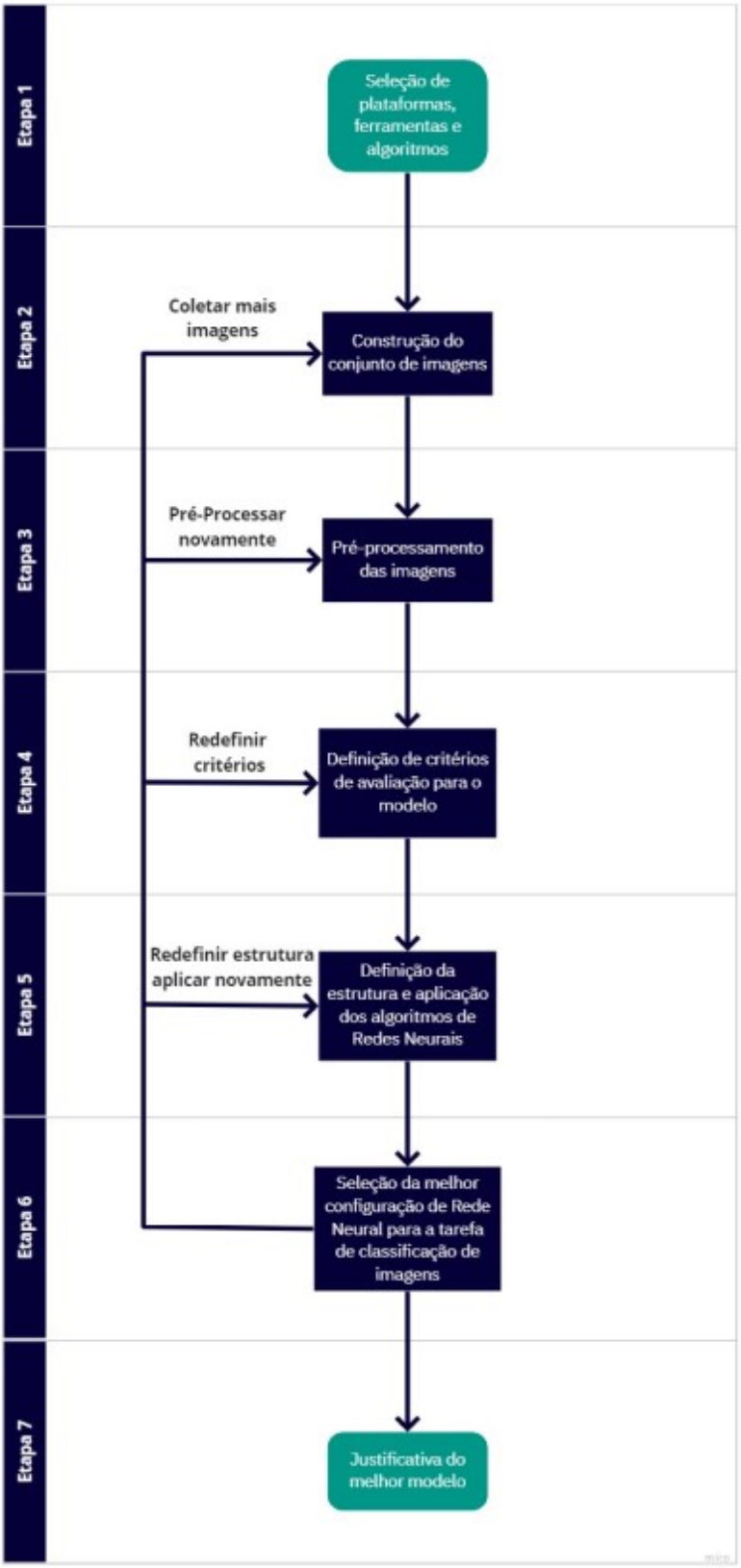


**Figura 31. Fluxograma das etapas de desenvolvimento do trabalho**


**Figura 32. Logotipos das ferramentas utilizadas**

Import TensorFlow

```
import tensorflow as tf

from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
```

**Figura 33. Captura de tela de trecho de código de tutorial do TensorFlow para importar biblioteca**
Obtido de https://www.tensorflow.org/tutorials/images/cnn. Acessado em: 16 jun. 2021

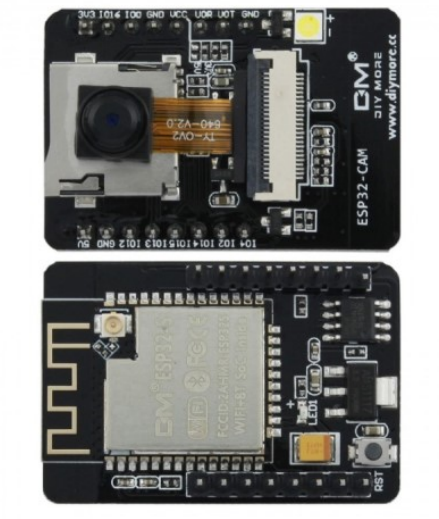

**Figura 34. ESP32-CAM**
Obtido de
https://www.usinainfo.com.br/esp32/esp32-cam-camera-ov2640-iot-5624.html.
Acessado em: 10 jun. 2021.

**Figura 35. Visão externa do protótipo de captura de imagens**


**Figura 36. Visão interna do protótipo de captura de imagens**

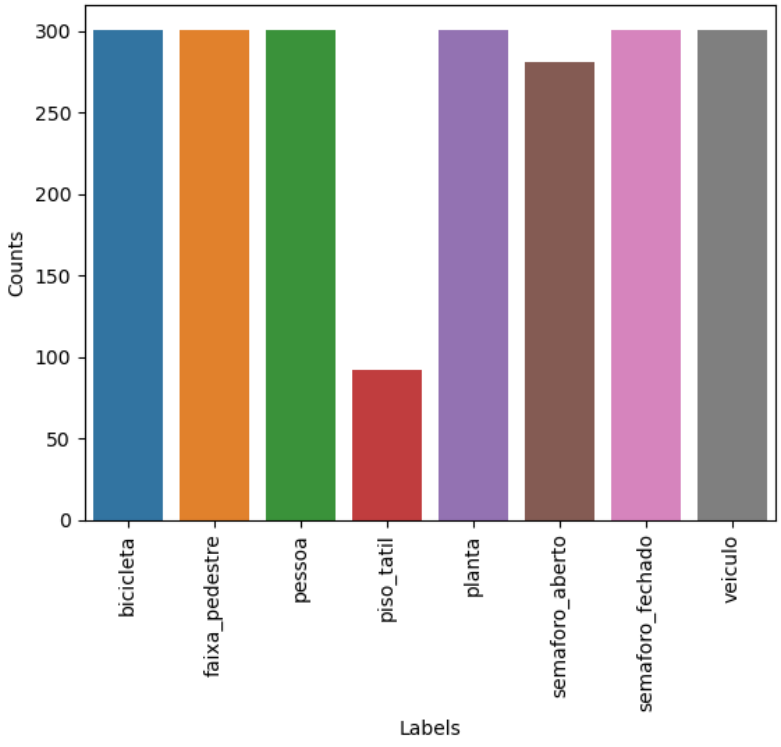


**Figura 37. Gráfico de amostras por classe**

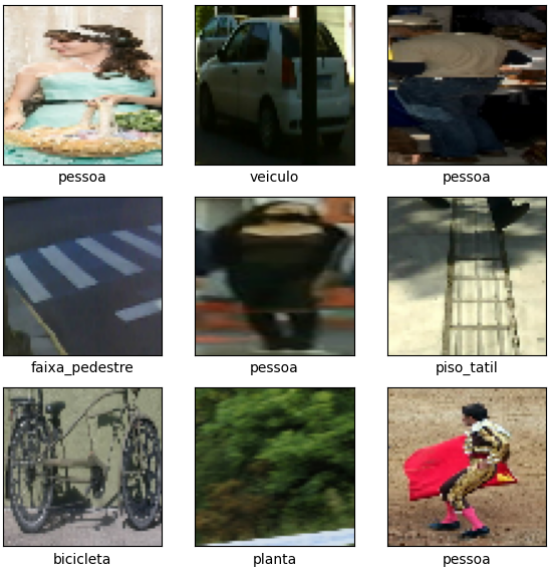


**Figura 38. Fragmento com algumas imagens do dataset**

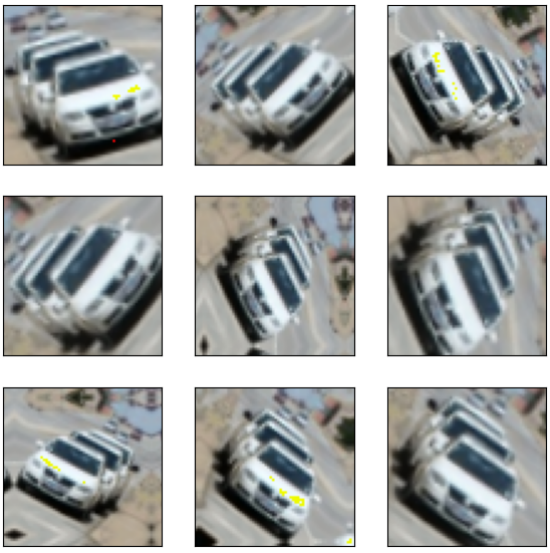

**Figura 39. Exemplo de uma imagem com augmentation**

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

**Figura 40. Captura de tela de trecho de código de tutorial do ThensorFlow para criação da estrutura de um modelo**

Obtido de https://www.tensorflow.org/tutorials/images/cnn. Acessado em: 16 jun. 2021.

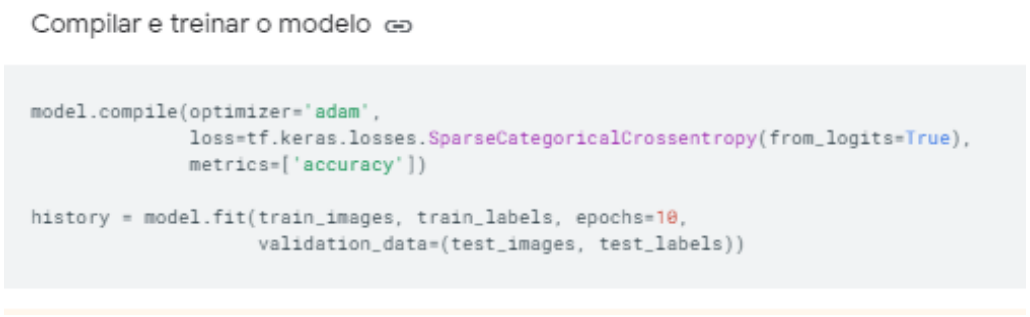


**Figura 41. Captura de tela de trecho de código de tutorial do ThensorFlow para compilar e treinar modelo**

Obtido de https://www.tensorflow.org/tutorials/images/cnn. Acessado em: 16 jun. 2021.

```
# K-fold Cross Validator
kfold = KFold(n_splits=n_folds, shuffle=True)


# K-fold Cross Validation Counter
fold_no = 1


acc_per_fold = []
loss_per_fold = []
histories = []

imagens = np.concatenate([x for x, y in dataset], axis=0)
labels = np.concatenate([y for x, y in dataset], axis=0)



for train_index, test_index in kfold.split(imagens, labels):
```

**Figura 42. Trecho de código que realiza o Cross-Validation**

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 62, 62, 32)        896
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 29, 29, 64)        18496
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 12, 12, 64)        36928
_________________________________________________________________
flatten_1 (Flatten)          (None, 9216)              0
_________________________________________________________________
dense_2 (Dense)              (None, 2048)              18876416
_________________________________________________________________
dense_3 (Dense)              (None, 8)                 16392
=================================================================
Total params: 18,949,128
Trainable params: 18,949,128
Non-trainable params: 0
_________________________________________________________________
```

**Figura 43. Estrutura do melhor modelo**

```
------------------------------------------------------------------------
Score per fold
------------------------------------------------------------------------
> Fold 1 - Loss: 1.2276252508163452 - Accuracy: 85.27919054031372%
------------------------------------------------------------------------
------------------------------------------------------------------------
> Fold 2 - Loss: 1.6308737993240356 - Accuracy: 80.71066141128854%
------------------------------------------------------------------------
------------------------------------------------------------------------
> Fold 3 - Loss: 1.8712540864944458 - Accuracy: 83.16326737403870%
------------------------------------------------------------------------
------------------------------------------------------------------------
> Fold 4 - Loss: 0.9532687067985535 - Accuracy: 85.71428656578064%
------------------------------------------------------------------------
------------------------------------------------------------------------
> Fold 5 - Loss: 0.8204505443572998 - Accuracy: 84.18367505073547%
------------------------------------------------------------------------
------------------------------------------------------------------------
> Fold 6 - Loss: 1.0997334718704224 - Accuracy: 85.20408272743225%
------------------------------------------------------------------------
------------------------------------------------------------------------
> Fold 7 - Loss: 1.8665887117385864 - Accuracy: 85.20408272743225%
------------------------------------------------------------------------
------------------------------------------------------------------------
> Fold 8 - Loss: 1.6211761236190796 - Accuracy: 82.14285969734192%
------------------------------------------------------------------------
------------------------------------------------------------------------
> Fold 9 - Loss: 1.0081024169921875 - Accuracy: 84.69387888908386%
------------------------------------------------------------------------
------------------------------------------------------------------------
> Fold 10 - Loss: 1.536873698234558 - Accuracy: 80.61224222183228%
------------------------------------------------------------------------

Average scores for all folds:
> Acurácia: 83.69082272052765 (+- 0.0182974803015205)
> Loss: 1.3635946810245514
------------------------------------------------------------------------
```

**Figura 44. Acurácias das partições e acurácia média**

Acurácia: 78.3410138248848 %

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.75 | 0.56 | 0.64 | 27 |
| 1 | 0.91 | 0.83 | 0.87 | 24 |
| 2 | 0.67 | 0.63 | 0.65 | 38 |
| 3 | 0.83 | 0.83 | 0.83 | 6 |
| 4 | 0.59 | 1.00 | 0.74 | 27 |
| 5 | 0.96 | 0.96 | 0.96 | 24 |
| 6 | 1.00 | 0.94 | 0.97 | 36 |
| 7 | 0.76 | 0.63 | 0.69 | 35 |
| accuracy | | | 0.78 | 217 |
| macro avg | 0.81 | 0.80 | 0.79 | 217 |
| weighted avg | 0.80 | 0.78 | 0.78 | 217 |

**Figura 45. Scores das predições**

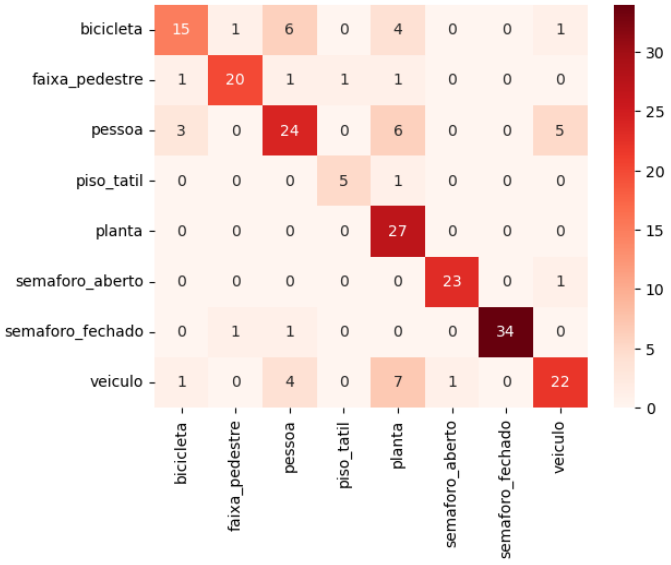


**Figura 46. Matriz de confusão das predições**

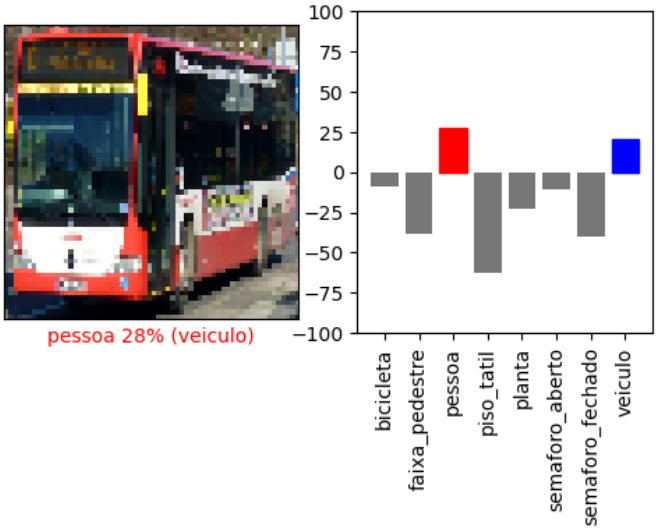


**Figura 47. Exemplo de predição parcialmente correta**