FUNDAÇÃO EDUCACIONAL DE FERNANDÓPOLIS
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

APLICATIVO MOBILE DE DETECÇÃO DE OBSTÁCULOS COM REDES NEURAIS CONVOLUCIONAIS

MOBILE OBSTACLE DETECTION APP WITH CONVOLUTIONAL NEURAL NETWORKS

Bruno César Duran[1], Igor Bianco Buosi[2], Jefferson Antonio Ribeiro Passerini[3]

[1] Fundação Educacional de Fernandópolis, brunoduran@fef.edu.br
[2] Fundação Educacional de Fernandópolis, igorbuosi@fef.edu.br
[3] Fundação Educacional de Fernandópolis, jefferson.passerini@fef.edu.br

Área: Informação e Comunicação.
Subárea: Matemática e Inteligência Computacional

RESUMO

Este projeto desenvolveu um aplicativo móvel para auxiliar pessoas com deficiência visual na detecção de obstáculos durante a locomoção, respondendo à necessidade de promover maior autonomia e segurança. Utilizando a câmera do smartphone, o sistema captura imagens do trajeto, processa-as localmente com a rede neural convolucional MobileNetV2 e classifica os dados por meio de um classificador MLP (*Multilayer Perceptron*). O feedback sonoro alerta o usuário sobre a presença ou ausência de obstáculos. O treinamento do modelo empregou a técnica de transferência de aprendizado com a base ImageNet, resultando em uma acurácia de 94,26% em cenários controlados e de 92,00% em condições reais, mesmo com variações de iluminação e tipos de obstáculos. Durante os testes, o sistema demonstrou equilíbrio entre precisão (91,42%) e sensibilidade (94,39%), confirmando sua capacidade de identificar trajetos seguros e obstáculos com eficiência. Os resultados evidenciam o potencial do aplicativo como um complemento à bengala, sem a necessidade de dispositivos adicionais ou conexão à internet. A inovação proporcionou maior autonomia ao usuário e superou o desempenho de estudos prévios, aumentando em 22,18% a acurácia. Assim, o modelo se apresenta como uma solução acessível e eficaz para promover a inclusão e a qualidade de vida de pessoas com deficiência visual.
Palavras-chave: Detecção de obstáculos. Redes neurais convolucionais. Transferência de aprendizado. Aprendizado supervisionado. Aplicativo *mobile*.

ABSTRACT

This project developed a mobile application to assist visually impaired individuals in obstacle detection during locomotion, addressing the need to promote greater autonomy and safety. Using the smartphone camera, the system captures images of the path, processes them locally with the MobileNetV2 convolutional neural network, and classifies the data using an MLP (Multilayer Perceptron) classifier. Audio feedback alerts the user about the presence or absence of obstacles. The model's training employed the transfer learning technique with the ImageNet dataset, achieving an accuracy of 94.26% in controlled scenarios and 92.00% in real-world conditions, even with variations in lighting and obstacle types. During testing, the system demonstrated a balance between precision (91.42%) and sensitivity (94.39%), confirming its capability to efficiently identify safe paths and obstacles. The results highlight the app's potential as a complement to the white cane, without the need for additional devices or an internet connection. This innovation provided users with greater autonomy and surpassed the performance of previous studies, increasing accuracy by 22.18%. Thus, the model emerges

as an accessible and effective solution to promote inclusion and improve the quality of life for visually impaired individuals.
Keywords: *Obstacle detection. Convolutional neural networks. Transfer learning. Supervised learning. Mobile app.*

1 INTRODUÇÃO

Atualmente, a deficiência visual, incluindo a cegueira total, é um tema significativo na sociedade global. As estimativas mais recentes indicam que pelo menos 2,2 bilhões de pessoas ao redor do mundo possuem algum tipo de deficiência visual, com aproximadamente 1 bilhão desses casos podendo ser prevenidos ou ainda não tratados.

Segundo dados da Organização Mundial da Saúde (OMS) de 2019, globalmente, 39 milhões de pessoas são cegas e 246 milhões possuem deficiência visual moderada a severa. No Brasil, os dados mais recentes ainda são do Censo de 2010, que registrou mais de 35 milhões de pessoas com algum grau de deficiência visual. Desses, 506.377 eram totalmente cegos. A proporção de pessoas com deficiência visual aumenta com a idade, afetando principalmente aqueles acima dos 50 anos devido a condições como a degeneração macular relacionada à idade.

As tecnologias assistivas têm evoluído para melhorar a acessibilidade e inclusão de pessoas com deficiência visual. Atualmente, aplicativos e dispositivos podem auxiliar na mobilidade urbana e acesso à informação, usando diferentes recursos para indicar obstáculos.

Portanto, no contexto presente, prevê-se a criação de um sistema que capacite pessoas com deficiência visual a utilizarem os recursos de seus *smartphones* para tarefas do dia a dia, ajudando na locomoção. Esse sistema funcionaria capturando imagens do trajeto que o usuário está percorrendo e fornecendo uma resposta por meio de áudio para alertar sobre obstáculos à frente, antes que estes sejam detectados pela bengala.

Esse novo sistema seria baseado em um projeto já desenvolvido no artigo “Auxílio a Deficientes Visuais Utilizando Redes Neurais Convolucionais”, desenvolvido por Sanga, Polo e Passerini (2023), entretanto agora sem necessidade de dispositivos adicionais ou conexão à internet, usando um classificador e uma rede convolucional diferente. O processamento será feito no próprio *smartphone* do usuário, sendo assim espera-se que o tempo de resposta seja menor e com uma precisão maior, tornando-o mais eficaz.

2 REFERENCIAL TEÓRICO

Esta seção tem como propósito introduzir os conceitos essenciais para compreender a metodologia proposta. Serão abordados temas como classificação das deficiências visuais e aprendizado profundo, especialmente focando em redes neurais convolucionais (CNN). Além disso, serão discutidos outros elementos como as ferramentas utilizadas.

2.1 CLASSIFICAÇÃO DE NÍVEIS DE DEFICIÊNCIA VISUAL

O Hospital de Olhos (2024) explica que o grau de acuidade visual mensura a capacidade funcional da visão de uma pessoa. Assim, quanto mais facilmente e com menos dificuldades alguém completa o teste de visão, maior é sua habilidade para enxergar e, consequentemente, melhor é sua acuidade visual. E vice-versa.

Conforme a 11ª Classificação Estatística Internacional de Doenças e Problemas Relacionados com a Saúde (CID-10), utiliza-se a acuidade visual do melhor olho para classificar a perda visual. Assim, define-se: ausência de deficiência visual ou deficiência visual leve (categoria 0) quando a acuidade é igual ou superior a 0,3; deficiência visual moderada (categoria 1) quando a acuidade é inferior a 0,3 e igual ou superior a 0,1; deficiência visual

grave (categoria 2) quando a acuidade é menor que 0,1 e igual ou superior a 0,05; cegueira (categoria 3) quando a acuidade é menor que 0,05 e igual ou superior a 0,02; cegueira (categoria 4) quando a acuidade é menor que 0,02 e maior ou igual à percepção de luz; e cegueira (categoria 5) quando não há percepção de luz.

2.2 APRENDIZADO PROFUNDO

De acordo com o Google (2024), o termo "inteligência artificial" (IA) refere-se a um conjunto de tecnologias projetadas para capacitar computadores a realizar uma variedade de tarefas complexas. Entre essas funções estão a habilidade de processar e compreender idiomas falados e escritos, interpretar informações visuais, analisar grandes volumes de dados e fornece recomendações personalizadas, entre outras aplicações. Essencialmente, a IA combina a ciência da computação com vastos conjuntos de dados, viabilizando atividades que, anteriormente, necessitavam da atuação humana direta. Além disso, o campo da IA inclui áreas específicas, como o Aprendizado de Máquina (*Machine Learning*) e o Aprendizado Profundo (*Deep Learning*).

Machine learning é uma subárea da inteligência artificial (IA) focada no desenvolvimento de sistemas capazes de aprender ou aprimorar seu desempenho com base nos dados que consomem (Oracle, 2024). Em geral, os algoritmos de aprendizado de máquina são aplicados para realizar previsões ou classificações, partindo de dados de entrada, que podem ser tanto rotulados quanto não rotulados. O algoritmo então, gera uma estimativa sobre um padrão existente nos dados. Uma função de erro é utilizada para avaliar a capacidade preditiva do modelo, se existirem exemplos conhecidos, essa função pode compará-los para determinar a precisão do modelo.

Atualmente, predominam dois principais tipos de algoritmos de *machine learning*: o aprendizado supervisionado e o aprendizado não supervisionado, diferenciando-se principalmente pelo método como cada um processa os dados para gerar previsões (Oracle, 2024).

No aprendizado supervisionado, que é o mais comum, um cientista de dados atua como um tutor, orientando o algoritmo sobre as inferências que ele deve fazer. De maneira similar a uma criança que aprende a reconhecer frutas através de um livro de imagens, o algoritmo é capacitado usando um conjunto de dados previamente rotulados com resultados específicos.

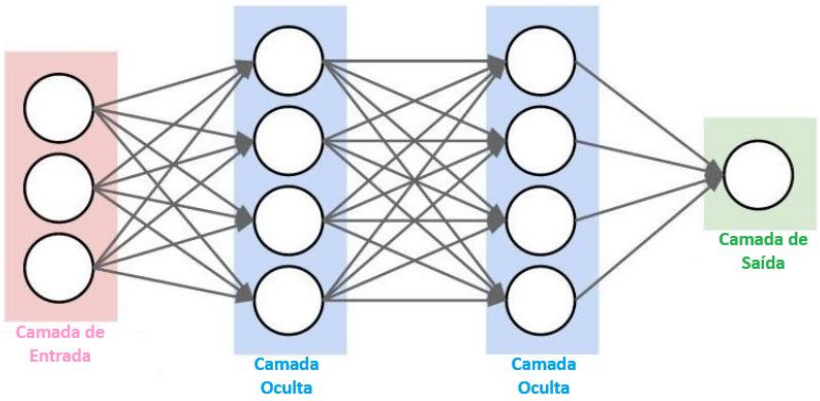
Por outro lado, o aprendizado de máquina não supervisionado adota uma abordagem autônoma, onde o computador aprende a discernir padrões e processos complexos sem a necessidade de supervisão humana contínua e direta. Este tipo de aprendizado se baseia em dados que não estão rotulados e não possuem uma saída específica estabelecida. Continuando com a analogia do aprendizado infantil, seria como uma criança aprendendo a identificar frutas observando suas cores e padrões, sem a ajuda de um professor para memorizar os nomes. A criança agruparia imagens semelhantes, atribuindo a cada grupo uma nova etiqueta própria.

Deep Learning é uma subárea do *Machine learning*, de acordo com a AWS (2024) o *DL* capacita computadores a processar dados de maneira semelhante ao cérebro humano, através de redes neurais. Por meio de modelos de aprendizagem profunda, é possível identificar padrões complexos em diversos tipos de dados, como imagens, texto e áudio, resultando em percepções e previsões precisas. Essa tecnologia permite automatizar tarefas que tradicionalmente requerem intervenção humana, como a descrição de imagens ou a transcrição de áudio em texto.

Segundo a AWS (2024), rede neural representa uma técnica de inteligência artificial que capacita computadores a analisar dados de maneira semelhante ao cérebro humano. Ela desenvolve um sistema adaptável que permite aos computadores aprenderem com erros e melhorar-se de forma contínua. As redes neurais artificiais são empregadas para resolver questões complexas, como o resumo de documentos ou a identificação precisa de imagens.

Uma rede neural profunda é composta por camadas de neurônios artificiais. Existem três tipos principais de camadas: camada de entrada, camadas ocultas e camada de saída. Camada de entrada é onde os dados brutos são recebidos, por exemplo, os pixels de uma imagem. Os dados da camada de entrada são processados e enviados para camadas mais profundas dentro da rede, chamadas de camadas ocultas, onde tratam as informações em diversos níveis e ajustam seu comportamento com base nas novas informações recebidas. Uma rede de aprendizado profundo pode ter centenas dessas camadas ocultas, permitindo a análise de um problema sob múltiplas perspectivas. E a camada de saída onde é fornecido o resultado do processamento, como a classificação de uma imagem.

Figura 1 – Representação visual das camadas de uma *Deep Learning*



Fonte: Elaborada pelos autores.

Os algoritmos de *deep learning* foram desenvolvidos para aprimorar a eficácia das abordagens convencionais de *machine learning*. Esses métodos convencionais requerem um significativo empenho humano para o treinamento do software.

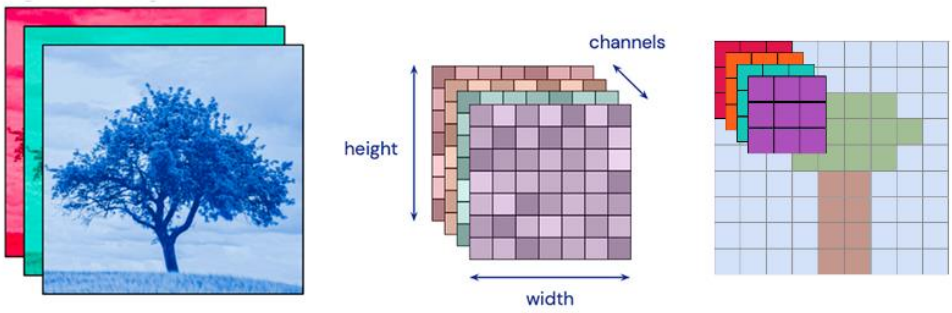
2.3 REDES NEURAIS CONVOLUCIONAIS

As Redes Neurais Convolucionais (CNNs) transformaram profundamente a visão computacional, particularmente na classificação de imagens. Elas possuem a habilidade de extrair características diretamente de pixels brutos, o que lhes permitiu obter êxitos impressionantes em diversos usos, como no reconhecimento de objetos, identificação facial e na avaliação de imagens médicas. As CNNs são projetadas para trabalhar com dados estruturados em forma de grade. Essas redes são formadas por diversas camadas, cada um desempenhando funções distintas.

Na camada de entrada é onde os valores brutos dos pixels de uma imagem são recebidos, marcando o começo da rede. Um pixel é basicamente um ponto em uma grade de imagem que possui uma cor específica, essa cor é representada por valores que combinam vermelho, verde e azul (RGB), cada um variando em intensidade. Sendo assim, somente com os dados brutos da imagem é muito difícil abstrair alguma informação que ajudará na classificação e/ou identificação da imagem.

Com o intuito de extrair características das imagens, as camadas convolucionais aplicam filtros (conhecimento também como kernels) sobre a imagem inicial. Esses filtros são adaptados para reconhecer padrões específicos, tais como bordas, cantos, texturas, cores através de uma operação chamada convolução. Nessa operação, o filtro move-se sobre a imagem, efetuando multiplicações e somas pontuais para criar mapas de características.

Figura 2 – Representação visual da camada convolucional



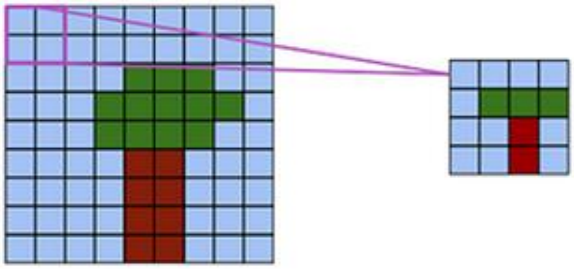
Fonte: Adaptado de UCL x DeepMind (2020).

É possível também recursos mais sofisticados após várias camadas convolucionais serem empilhadas camada por camada (Jia Song, Shaohua Gao, Yunqiang Zhu & Chenyan Ma 2019).

Após extrair as características, é introduzido elementos não-lineares à rede através das funções de ativação, isso capacita a rede a aprender padrões complexos. Entre as funções de ativação mais usadas em *CNNs* estão a Unidade Linear Retificada (ReLU), a função sigmoidal e a tangente hiperbólica.

Nas camadas de agrupamento (*Pooling*) ocorre a compressão das dimensões espaciais dos mapas de características produzidos pelas camadas convolucionais. Por exemplo, o agrupamento máximo escolhe o maior valor de cada área de agrupamento, diminuindo a dimensionalidade enquanto mantém os aspectos mais relevantes.

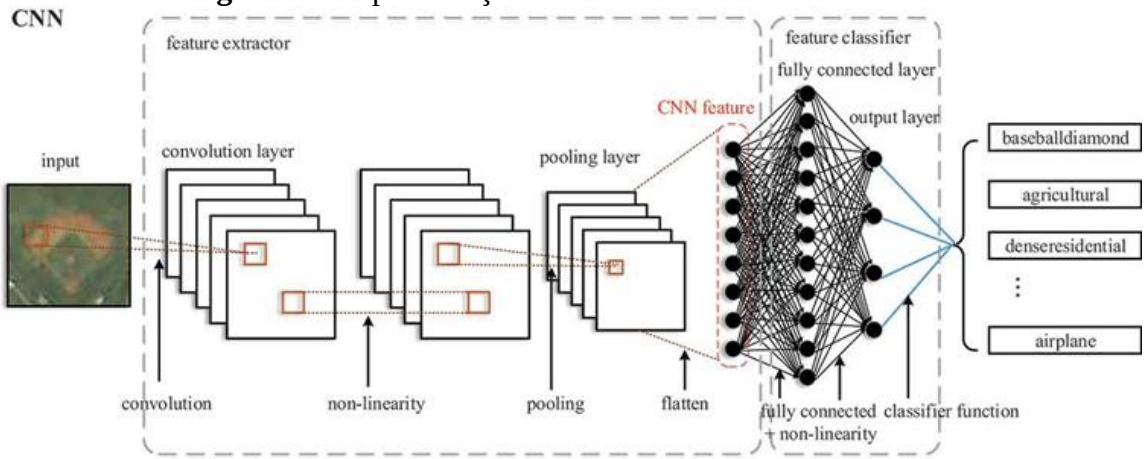
Figura 3 – Representação visual da camada de agrupamento (*Pooling*)



Fonte: UCL x DeepMind (2020).

Por fim, na camada de saída é apresentada as classificações ou previsões finais, com base nas representações que foram aprendidas pela rede.

Figura 4 – Representação visual das camadas de uma *CNN*



Fonte: Jia Song, Shaohua Gao, Yunqiang Zhu & Chenyan Ma (2019).

2.4 TRANSFERÊNCIA DE APRENDIZADO *(TRANSFER LEARNING)*

A transferência de aprendizado é uma técnica em *machine learning* onde o conhecimento previamente obtido em uma tarefa ou modelo é utilizado para aprimorar o desempenho em outra tarefa ou modelo relacionado. Em vez de iniciar o treinamento de um novo modelo do zero, essa abordagem possibilita o uso de um modelo pré-treinado (geralmente desenvolvido com uma grande quantidade de dados) como ponto de partida, adaptando suas camadas finais para a tarefa específica desejada.

A IBM (2024) afirma que a transferência de aprendizado diminui os custos computacionais necessários para desenvolver modelos voltados para novos problemas, uma vez que permite o uso de conjuntos de dados menores. A técnica consiste no retreinamento do modelo já existente com um novo conjunto de dados, fazendo com que o modelo atualizado integre conhecimentos adquiridos em múltiplos conjuntos de dados. Assim, o modelo retreinado pode alcançar um desempenho superior e generalizar melhor ao ser exposto a uma variedade de dados, em comparação ao modelo inicial, que foi treinado em um único conjunto de dados. Dessa forma, a transferência de aprendizado também contribui para reduzir o risco de *overfitting*.

2.5 TENSORFLOW

De acordo com a documentação oficial, o TensorFlow, uma iniciativa do Google, é uma plataforma de *machine learning* de código aberto que tem se destacado pela sua robustez e versatilidade em diversos ambientes computacionais, desde dispositivos móveis até servidores. Essa ferramenta é projetada para facilitar o desenvolvimento e treinamento de modelos de *machine learning*, proporcionando aos usuários, independentemente do nível de experiência, a capacidade de lidar com volumes significativos de dados.

Essencialmente, a plataforma é reconhecida por sua flexibilidade na experimentação de arquiteturas de redes neurais, com um desempenho que não compromete a velocidade graças ao suporte ao treinamento distribuído. Isso permite uma otimização eficaz dos recursos e a minimização dos tempos de treinamento, o que é crucial para a aplicação prática de modelos de aprendizado de máquina em ambientes de produção.

Além das capacidades técnicas, o TensorFlow também enfatiza a importância das práticas responsáveis de IA. A plataforma inclui ferramentas integradas para o diagnóstico e ajuste de modelos, promovendo a transparência e a ética na implementação de soluções de IA. Esses recursos asseguram que os modelos não apenas atendam aos padrões de desempenho, mas também se alinhem com os princípios de uso responsável da tecnologia.

2.6 KERAS

Keras é uma API de alto nível integrada ao TensorFlow, desenvolvida em código aberto com a linguagem Python, e voltada para o desenvolvimento e treinamento de modelos de aprendizado profundo (*deep learning*). Ela simplifica a criação de redes neurais, proporcionando uma experiência que facilita a prototipagem rápida de modelos avançados sem a necessidade de muitos detalhes técnicos de implementação.

2.7 MLP *(MULTILAYER PERCEPTRON)*

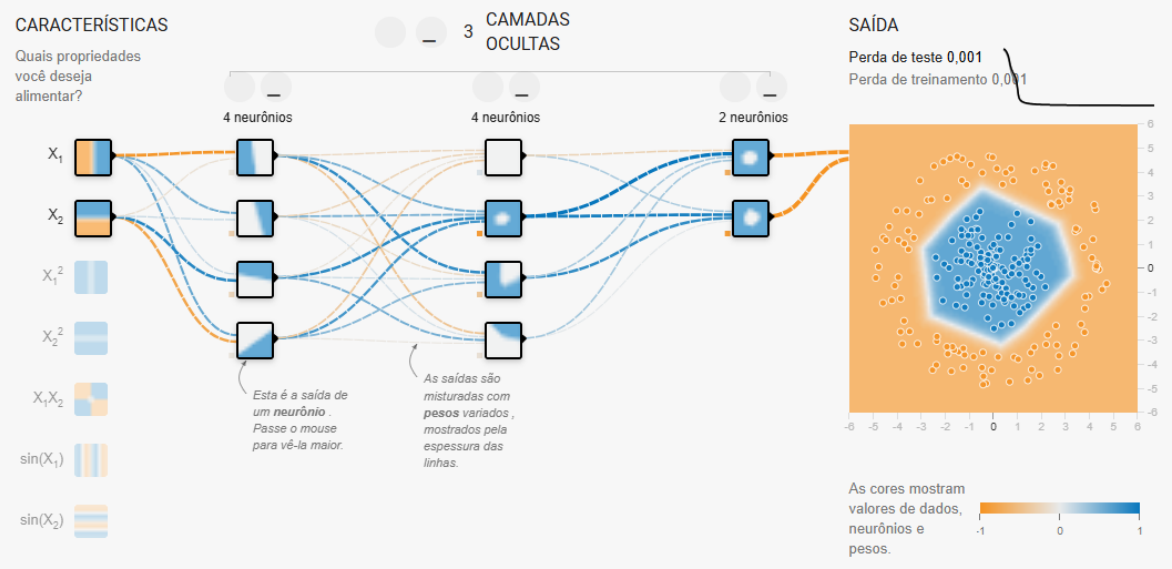
Conforme Jaiswal (2024), o *perceptron* multicamada (MLP, ou *multilayer perceptron*) é um modelo de rede neural artificial composto por múltiplas camadas de neurônios

interconectados. Cada neurônio do MLP emprega, em geral, funções de ativação não lineares, o que possibilita à rede a capacidade de identificar padrões complexos dentro dos dados. Devido a essa característica, os MLPs desempenham um papel crucial em aprendizado de máquina, sendo especialmente úteis para captar relações não lineares em diferentes conjuntos de dados, tornando-se, assim, modelos robustos para tarefas como classificação, regressão e reconhecimento de padrões.

Cada neurônio de uma camada do MLP se conecta a todos os neurônios da camada seguinte, configurando uma rede densa onde cada ligação possui um peso específico. Durante o processo de treinamento, a rede ajusta esses pesos, buscando minimizar o erro em suas previsões.

MLPs têm sido amplamente aplicados em áreas diversas, como reconhecimento de imagem, processamento de linguagem natural e reconhecimento de fala. A flexibilidade em sua arquitetura e a habilidade de aproximar praticamente qualquer função, sob determinadas condições, fazem do MLP uma estrutura essencial para o desenvolvimento do aprendizado profundo e pesquisas em redes neurais.

Figura 5 – Representação visual de um MLP com saída não linear



Fonte: Tensorflow - neural network playground (2024).

2.8 FIREBASE

O Firebase é uma plataforma desenvolvida pelo Google para o desenvolvimento de aplicativos móveis e web. Trata-se de um serviço de *back-end as a service* (BaaS, ou *back-end* como serviço) que oferece uma série de funcionalidades de *back-end*, como banco de dados em tempo real e armazenamento de arquivos. A principal vantagem do Firebase é permitir que desenvolvedores integrem recursos de *back-end* sem a necessidade de se preocupar com a criação e manutenção de toda a infraestrutura envolvida.

2.9 ANDROID

De acordo com Deitel (2015), o Android é um dos sistemas operacionais mais reconhecidos para dispositivos móveis e foi inicialmente criado pela *Android Inc*. Em 2005, a Google comprou a empresa e assumiu o comando de seu desenvolvimento. Em 2007, a Google formou a Open Handset Alliance com várias outras empresas tecnológicas. O objetivo da

aliança era desenvolver, manter e aprimorar o Android, buscando constantemente inovações e melhorias na experiência dos usuários e também a redução de custos.

É fundamental mencionar sua plataforma de desenvolvimento, o *Kit* de Desenvolvimento de Programas (*SDK*). Esse kit fornece todas as ferramentas necessárias para criar aplicativos e, atualmente, a versão mais recente é do Android 13, API nível 33, segundo Developers Android (2023). Quanto ao desenvolvimento de aplicativos, de acordo com Developers Android (2019), eles podem ser programados em Kotlin, que é a linguagem oficialmente preferida, além de *Java* e C++, oferecendo três opções de linguagens para desenvolvedores.

No sistema operacional *Android*, cada aplicativo opera dentro de um espaço de segurança isolado conhecido como *Sandbox*, reforçado por várias medidas de segurança. O Android usa uma base Linux multiusuário, onde cada aplicativo é tratado como um usuário distinto com um código de usuário do Linux exclusivo, garantindo que os arquivos de um aplicativo só sejam acessíveis por ele. Adicionalmente, cada aplicativo roda em sua própria máquina virtual (VM), mantendo seu código isolado de outros aplicativos. Os processos dos aplicativos são gerenciados pelo Android, que os inicia conforme necessário e os encerra para liberar memória ou quando não são mais necessários.

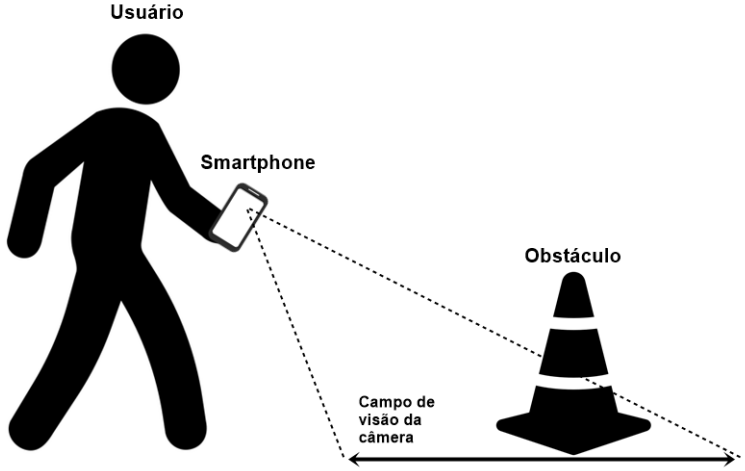
3 METODOLOGIA

Nesta seção, descreve-se minuciosamente o funcionamento do projeto e as tecnologias desenvolvidas para oferecer suporte a pessoas com deficiência visual. Explicam-se também os métodos empregados para a classificação de imagens do trajeto à frente do smartphone, identificando se há ou não obstáculos.

Baseamo-nos no estudo realizado por Sanga, Polo e Passerini (2023), que obteve uma acurácia média de 75,3% ao aplicar uma rede neural convolucional com técnica de *transfer learning*, usando a base ImageNet, no caso, uma combinação dos modelos VGG16 e VGG19, juntamente com um classificador SVM.

A figura 6 representa a utilização por um usuário do aplicativo, onde o sistema emitirá alertas de áudio no smartphone com aviso antecipado a obstáculos.

Figura 6 – Representação visual da utilização do aplicativo



Fonte: Os autores (2024).

O aplicativo utiliza a câmera do smartphone para capturar imagens de forma contínua e emite alertas sonoros quando detecta ou não a presença de obstáculos. Todo o processamento ocorre localmente no dispositivo, usando um modelo de aprendizado de máquina otimizado no

formato TensorFlow Lite, uma versão do TensorFlow desenvolvida especificamente para dispositivos móveis e com menor capacidade de processamento, como smartphones.

A tecnologia aplicada envolve uma rede neural convolucional (CNN), que foi treinada com a base de dados ImageNet usando *transfer learning*. Para esse caso, foi utilizado a MobileNetV2, um modelo adequado para dispositivos móveis, com um MLP (*Multilayer Perceptron*) utilizado como classificador. Esse MLP foi previamente treinado com imagens livres e de diversos tipos de obstáculos. Resumidamente, o fluxo de trabalho consiste em uma aplicação desktop que extrai as características das imagens usadas no treinamento, criando um modelo que é posteriormente transferido para o aplicativo móvel. Dessa forma, o aplicativo pode executar as análises e classificações sem depender de conexão à internet, funcionando de maneira independente.

Durante o uso, o modelo retornará ao usuário uma indicação sobre o caminho: emite um sinal sonoro único para indicar que o caminho está livre e dois sinais sonoros consecutivos quando identifica um obstáculo à frente.

3.1 TREINAMENTO DO MODELO

Detalhando-se o processo descrito no tópico anterior, o modelo de classificação será composto por duas etapas: extração de características e classificação dessas características utilizando o modelo de aprendizado supervisionado MLP (*Multilayer Perceptron*).

Inicialmente, na fase de preparação, as imagens da base de dados são organizadas e categorizadas em duas classes fundamentais: "*clear*" (sem obstáculo) e "*non-clear*" (com obstáculo). Essa categorização é baseada nos nomes dos arquivos, o que facilita a separação adequada das classes antes da extração de características. Essa etapa é crucial para garantir uma distinção clara entre as categorias, proporcionando uma base sólida para a fase de aprendizado supervisionado subsequente.

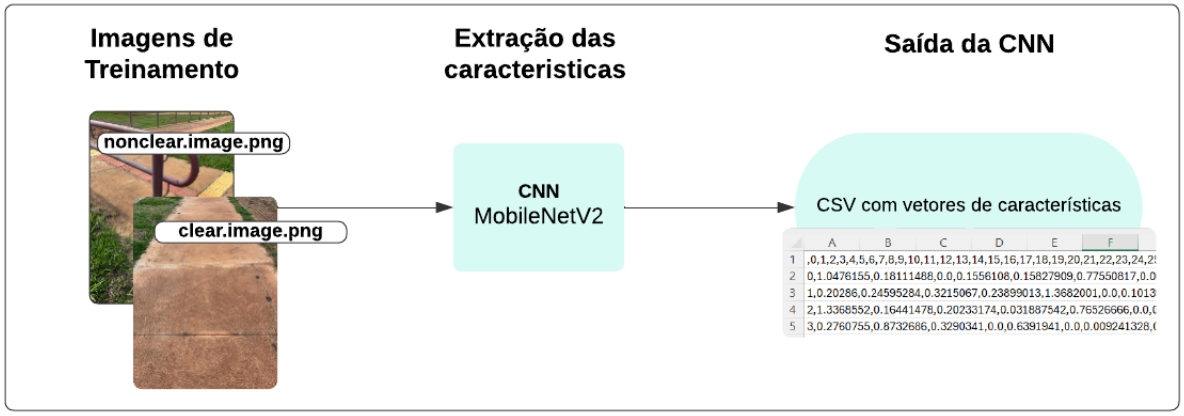
Durante a extração de características, emprega-se a técnica de aprendizado por transferência (*transfer learning*) utilizando o modelo MobileNetV2, uma rede neural convolucional pré-treinada no conjunto de dados ImageNet. A escolha desse modelo específico se deve à sua capacidade de capturar características visuais de forma eficiente, mesmo em contextos nos quais o conjunto de dados é limitado, além de ser projetada para dispositivos móveis e outras plataformas com restrições de recursos, como processamento e memória, que é nosso caso. Durante esse processo, todas as imagens são redimensionadas para 224x224 pixels, o que se alinha com o tamanho de entrada esperado pelo MobileNetV2, garantindo consistência no pré-processamento das imagens.

Após o redimensionamento, as imagens são submetidas a uma função de pré-processamento específica do MobileNetV2, chamada *preprocess_input*, que normaliza os valores dos pixels para o intervalo esperado pelo modelo. Esse ajuste inclui a subtração da média RGB dos valores de pixels do conjunto de dados ImageNet, reduzindo a variabilidade indesejada nas cores e tornando as imagens mais padronizadas para análise pela rede. Ao aplicar essa normalização, a função ajuda o modelo a focar em aspectos estruturais das imagens, como padrões de textura, formas e contornos, essenciais para o reconhecimento de objetos independentemente das condições de iluminação e cor das imagens originais. Esse ajuste específico de entrada, somado ao redimensionamento, garante que as imagens estejam no formato ideal para o MobileNetV2, maximizando a eficiência da extração de características.

Em seguida, as imagens passam pela última camada convolucional da rede, excluindo-se a camada totalmente conectada (*fully-connected*). Dessa maneira, o MobileNetV2 gera vetores de características que representam aspectos visuais relevantes das imagens, como texturas, formas e bordas, sem realizar uma classificação final. Esses vetores de características são então

exportados para um arquivo CSV, que forma a base de dados necessária para a etapa de treinamento do classificador de reconhecimento de obstáculos.

Figura 7 – Modelo geral da extração das características



Fonte: Os autores (2024).

Após a etapa de extração de características, inicia-se o treinamento do modelo de aprendizado supervisionado MLP (*Multilayer Perceptron*), utilizando as características previamente extraídas. Primeiramente, as *features* são carregadas do arquivo CSV gerado na etapa anterior. Com os dados em mãos, as classes são mapeadas para valores numéricos, com "*clear*" representado por 1 e "*non-clear*" por 0, facilitando a manipulação numérica necessária para o treinamento do modelo.

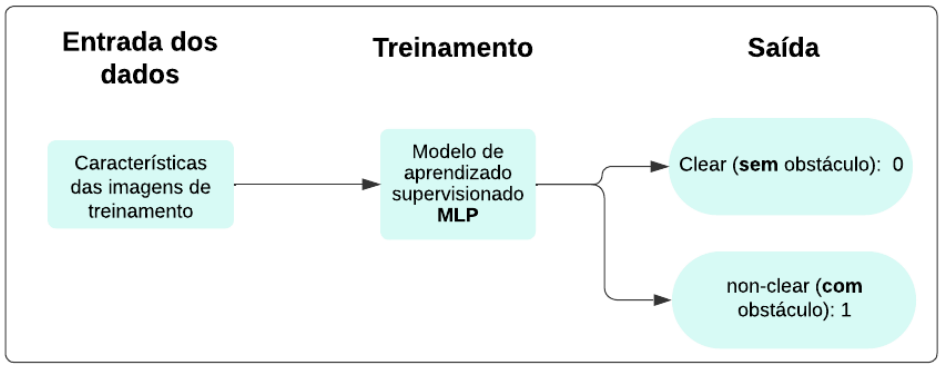
Em seguida, os dados são divididos em conjuntos de treino e validação, seguindo uma divisão de 80% para treino e 20% para validação. Essa separação é essencial para avaliar o desempenho do modelo em dados não vistos durante o treinamento, o que auxilia na detecção de *overfitting*. O *Overfitting* ocorre quando o modelo aprende excessivamente os detalhes e ruídos do conjunto de dados de treino, em detrimento de sua capacidade de generalização para novos dados, resultando em um desempenho inferior nos dados de teste (Goodfellow et al., 2016).

Com os dados preparados, o modelo MLP é configurado na biblioteca Keras com uma camada densa de 256 neurônios, utilizando a função de ativação ReLU. A função ReLU (*rectified linear unit*) ajusta os valores de entrada, zerando os negativos e preservando os positivos, o que permite ao modelo capturar relações não-lineares, agilizando o processo de treinamento. Para prevenir o *overfitting*, a camada é também regularizada com L2, uma técnica que penaliza pesos excessivamente altos no modelo, ajudando a equilibrar os parâmetros e reduzir a complexidade da rede. Além disso, uma camada *dropout* com taxa de 50% é utilizada, desativando aleatoriamente metade dos neurônios durante o treinamento, o que aumenta a robustez do modelo ao reduzir a dependência de neurônios específicos.

A última camada é uma camada de saída com ativação sigmoide, que realiza a classificação binária entre as classes "*clear*" e "*non-clear*". O modelo é então compilado com a função de perda binária *binary crossentropy*, que calcula a diferença entre a previsão do modelo e o rótulo real para cada amostra. O otimizador RMSprop, também é utilizado no treinamento, ajustando os pesos do modelo com base no gradiente do erro e aplicando uma média dos gradientes recentes para otimizar o processo de atualização dos pesos.

Para finalizar o treinamento do modelo denso, são aplicados *callbacks*, como *early stopping*, que interrompe o treinamento caso a perda de validação não melhore após cinco épocas/tentativas consecutivas, e uma função personalizada de matriz de confusão para registrar o desempenho em cada época.

Figura 8 – Modelo denso



Fonte: Os autores (2024).

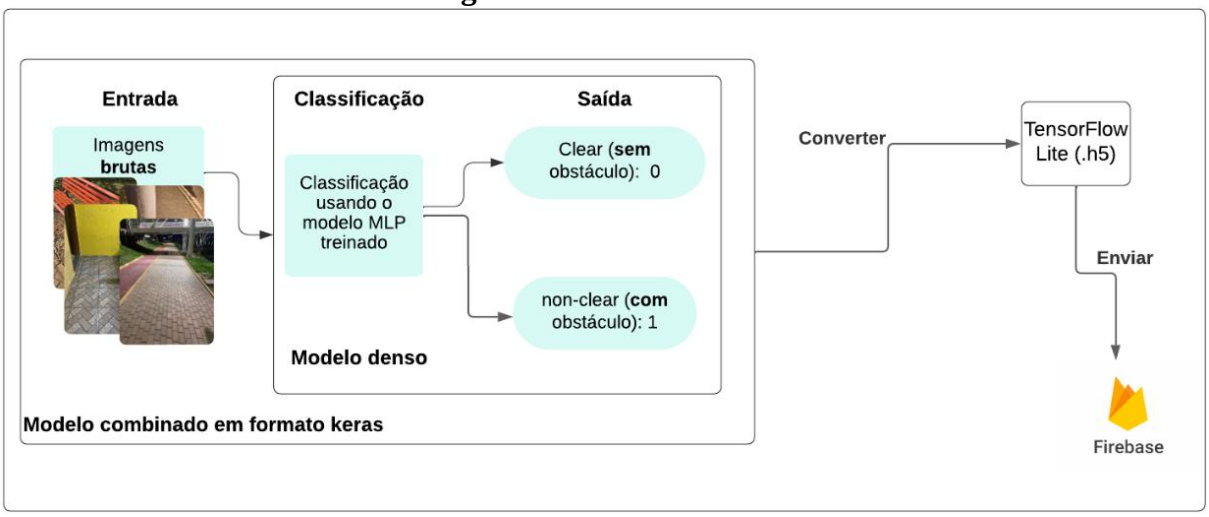
Após a etapa de treinamento do modelo denso, uma camada de entrada é adicionada para permitir que o modelo aceite diretamente imagens brutas para classificação. Esse processo é realizado combinando a rede MobileNetV2 com o modelo denso previamente treinado. Inicialmente, o modelo denso é carregado para integrar-se ao novo modelo combinado, em seguida, é criada uma entrada de imagem no formato adequado para o MobileNetV2 (224x224 pixels com 3 canais de cor).

O MobileNetV2, que serve como uma base de extração de características, é carregado com pesos pré-treinados e suas camadas são congeladas para manter as características visuais aprendidas no modelo denso, o que impede o ajuste durante o treinamento. A imagem de entrada passa primeiro pela MobileNetV2, onde suas características visuais são extraídas e achatadas. Essas características são então encaminhadas ao modelo denso já treinado, que realiza a classificação final entre as classes "*clear*" e "*non-clear*". Esse novo modelo combinado é compilado e salvo em formato Keras, e depois convertido para tensorflow lite, permitindo sua execução eficiente em dispositivos móveis.

Com esse modelo combinado, é possível realizar a classificação diretamente a partir de imagens, utilizando a capacidade do MobileNetV2 para extração automática de características visuais, seguida pela classificação com o modelo denso treinado, proporcionando uma solução completa para o reconhecimento de obstáculos em tempo real.

Após isso, o modelo em tensorflow lite é enviado para o Firebase Storage, permitindo que seja posteriormente baixado e utilizado pelo aplicativo Android. Isso facilita a distribuição e atualização do modelo, garantindo que o aplicativo tenha acesso à versão mais recente.

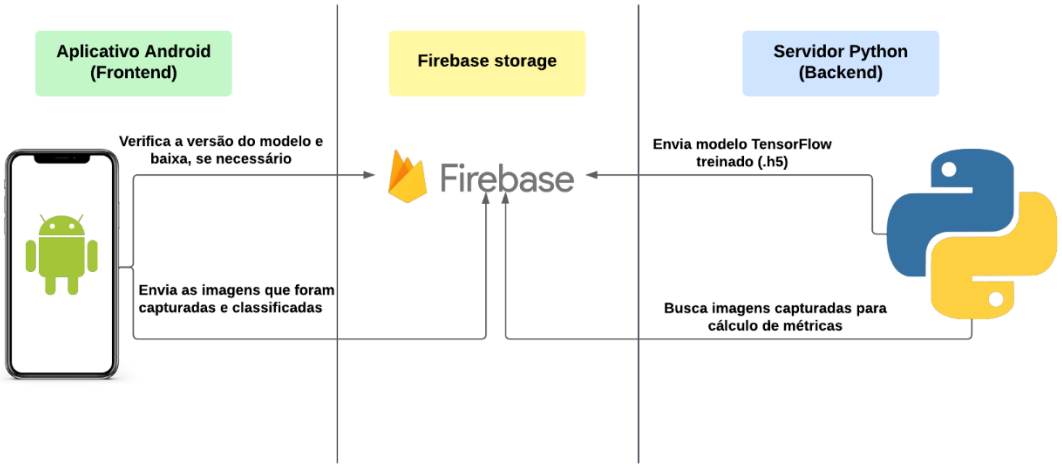
Figura 9 – Fluxo do modelo final



Fonte: Os autores (2024).

Essa arquitetura foi planejada para garantir um baixo acoplamento entre o *backend* e o *frontend*. O aplicativo Android não precisa de uma conexão direta com o servidor para funcionar, pois realiza apenas a consulta ao Firebase Storage para obter a versão mais recente do modelo treinado. Após o download, o modelo é armazenado localmente, permitindo que a classificação ocorra diretamente no dispositivo. Isso elimina a necessidade de conexão constante com a internet, já que o processamento e a classificação das imagens são realizados internamente, proporcionando uma solução eficiente e funcional mesmo em ambientes offline.

Figura 10 – Fluxo de comunicação entre aplicativo android, firebase e *backend*



Fonte: Os autores (2024).

3.2 APLICATIVO MOBILE

O modelo de aprendizado de máquina em formato TensorFlow Lite não está previamente incorporado ao aplicativo. Para facilitar o processo, foi criada uma função que envia o modelo para um armazenamento no Firebase, que foi detalhado anteriormente. O aplicativo conta com uma funcionalidade para baixar ou atualizar o modelo configurado, simplificando o processo sempre que um novo treinamento é realizado e um novo modelo é gerado.

Figura 11 – Tela de configuração do App, onde é possível sincronizar o modelo mais atualizado

Fonte: Os autores (2024).

Na Figura 11, destaca-se a tela de configurações, onde a opção “sincronizar” permite o download ou a atualização do modelo no dispositivo. Uma vez realizada a sincronização, o aplicativo estará preparado para realizar a classificação das imagens.

Ao pressionar o botão para iniciar a classificação, uma imagem é capturada pela câmera. Antes de ser processada pelo modelo, a imagem passa por uma etapa de normalização: primeiro, ela é rotacionada para garantir a orientação correta. Em seguida, é redimensionada para 224x224 pixels, correspondendo ao tamanho de entrada exigido pelo MobileNetV2. Após o redimensionamento, os valores dos pixels são ajustados para o intervalo esperado pelo modelo. Esse procedimento é idêntico ao utilizado na etapa de extração de características das imagens durante o treinamento do modelo.

Com a imagem normalizada, realizamos sua classificação utilizando o modelo *TensorFlow Lite* previamente configurado. Após a classificação, o aplicativo emite um sinal sonoro: um único *beep* indica que nenhum obstáculo foi detectado, enquanto dois *beeps* sinalizam a presença de um obstáculo.

Após o processamento da imagem, o aplicativo também a armazena internamente, nomeando-a como “*clear*” quando nenhum obstáculo é detectado, ou “*non-clear*” caso algum obstáculo seja identificado. Essas imagens serão utilizadas posteriormente para avaliar a precisão do modelo.

O processo se repete automaticamente: o aplicativo captura uma nova imagem com a câmera, realiza sua normalização, efetua a classificação, emite o sinal sonoro correspondente e armazena a imagem. Esse ciclo continua até que o usuário selecione a opção para interromper a classificação.

Na Figura 12, temos uma representação do funcionamento do aplicativo em situações real de navegação, juntamente da sua interface, que reage conforme o caminho. Na parte (a), observa-se a detecção de um trajeto livre de obstáculos, enquanto em (b), o aplicativo identifica a presença de um obstáculo no caminho.

Figura 12 – Demonstração da utilização do aplicativo em situação real

Fonte: Os autores (2024).

Para captar as imagens, a câmera do smartphone deve estar posicionada verticalmente e ligeiramente inclinada para o chão, de modo a maximizar o campo de visão e permitir uma leitura mais precisa do caminho adiante.

A partir da opção “sincronizar” no menu de configurações, destacado na Figura 11, além de baixar ou atualizar o modelo, também envia as imagens armazenadas internamente para o

Firebase Storage. Posteriormente, essas imagens são baixadas pelo *backend* para avaliar a precisão do modelo. Esse processo está ilustrado na Figura 10.

No mesmo menu, há uma opção chamada “limpar imagens”, que apaga as imagens armazenadas durante o processo. Essa funcionalidade foi implementada para situações de teste em que não se deseja enviar as imagens para o Firebase Storage. Além disso, existe uma opção para “enviar imagens automaticamente”. Quando ativada, enquanto o aplicativo captura e classifica as imagens, ele armazena um lote a cada dez imagens e as envia para o Firebase em segundo plano, permitindo que o processo de captura e classificação continue sem interrupções e agilize o envio das imagens.

Implementamos essa funcionalidade como uma opção para que dispositivos com menor capacidade de processamento não sofram perda de desempenho ao classificar imagens enquanto as enviam para o armazenamento externo. Caso seja necessário otimizar o desempenho, basta desativar essa opção.

3.3 MATERIAIS

Todos os desenvolvimentos foram feitos no sistema operacional Windows 11 de 64bits. O treinamento e exportação do modelo foi desenvolvida na linguagem Python, versão 3.11, executados na plataforma Anaconda onde foi utilizada a IDE PyCharm Community Edition na versão 2024.1.4. O aplicativo Android foi desenvolvido na linguagem Java, versão 11.0.16, utilizando a IDE Android Studio na versão 2024.1.

Para a captura de imagens, foi utilizado um smartphones Samsung Galaxy S10 Plus, modelo de 2019, com processador Exynos 9820 de frequência máxima de 2,7GHz, placa gráfica de 0.607 TFLOPS, 8GB de memória RAM, com sistema operacional Android 12 Samsung One UI 4.1.

4 ANÁLISE E DISCUSSÃO DOS RESULTADOS

As análises serão detalhadas com base em dois conjuntos de dados: os dados de treinamento, utilizados para treinar e ajustar o modelo, e os dados reais, que simulam cenários de aplicação prática. A comparação entre esses dois conjuntos permitirá avaliar o desempenho e a consistência do modelo tanto em condições controladas quanto em situações reais. Com isso, poderemos verificar se o modelo é capaz de generalizar bem e identificar obstáculos de maneira precisa em condições diferentes das observadas durante o treinamento.

Para avaliar a performance do modelo, foram empregadas várias métricas cruciais, cada uma responsável por medir aspectos específicos de seu desempenho. A época, por exemplo, corresponde a uma rodada completa de treinamento do modelo, enquanto a acurácia indica a proporção de previsões corretas, oferecendo uma visão geral da taxa de acertos. A pontuação F1 é útil em cenários com classes desbalanceadas, pois combina precisão e sensibilidade, proporcionando um equilíbrio entre esses dois aspectos.

A métrica ROC examina as taxas de verdadeiros positivos e falsos positivos sob diferentes configurações, e sua curva permite avaliar como o modelo se comporta em variados ajustes de sensibilidade. Outras métricas, como precisão e sensibilidade (ou *recall*), ajudam a entender o equilíbrio entre acertar as previsões (precisão) e identificar corretamente todos os obstáculos (*recall*). A especificidade, por sua vez, mede a capacidade do modelo de reconhecer corretamente a ausência de obstáculos, enquanto a taxa de falsos positivos (FPR) e a taxa de falsos negativos (FNR) revelam o quanto o modelo falha ao prever corretamente.

A matriz de confusão atua como uma ferramenta visual importante para analisar esses erros, destacando onde estão concentradas as principais falhas. Seus quadrantes, quantifica as seguintes informações: Verdadeiro Positivo (VP) - o modelo previu corretamente uma imagem

sem obstáculo; Falso Positivo (FP) - o modelo previu sem obstáculo quando havia obstáculo; Falso Negativo (FN) - o modelo previu com obstáculo quando não havia; Verdadeiro Negativo (VN) – o modelo previu corretamente uma imagem com obstáculo.

Em conjunto, essas métricas oferecem uma visão abrangente sobre a eficácia do modelo, permitindo que se determine sua precisão e confiabilidade.

4.1 RESULTADO DO MODELO

Foi utilizada uma base de imagens previamente desenvolvida por Sanga, Polo e Passerini (2023), com a inclusão de novas imagens produzidas pelos autores. A base totaliza 610 fotografias de caminhos, igualmente distribuídas entre aquelas que apresentam obstáculos e as que não apresentam.

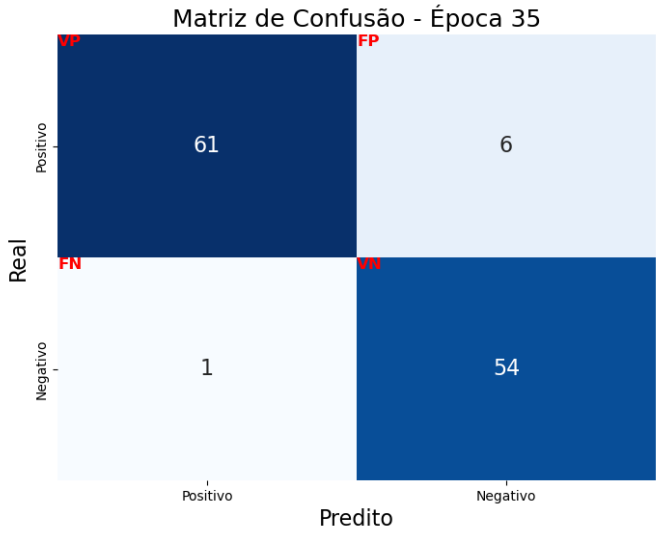
Com base nos resultados da última época (época 35), após o treinamento com 122 amostras (20% da base total de treinamento), o modelo alcançou uma acurácia de 94,26%, indicando que mais de 94% das previsões realizadas estavam corretas. A pontuação F1 de 94,57% reflete um bom equilíbrio entre precisão e sensibilidade.

A sensibilidade (*recall*) de 98,39% mostra que o modelo foi altamente eficaz em identificar quase todas as áreas sem obstáculos (verdadeiros positivos), enquanto a precisão de 91,04% indica que, entre as previsões de áreas sem obstáculos, a maioria estava correta. A especificidade de 90,00% sugere uma taxa sólida de acertos na identificação de áreas com obstáculos.

A taxa de falsos positivos (FPR) de 10% e a taxa de falsos negativos (FNR) de 1,61% indicam uma baixa margem de erro. O modelo apresenta uma leve tendência a "errar" mais ao prever que uma área está sem obstáculos, o que pode ser indesejável em alguns contextos de segurança, onde falhar ao identificar um obstáculo pode representar um risco.

A matriz de confusão representada na figura 13 confirma esses resultados, com 54 verdadeiros negativos (imagens com obstáculos), 61 verdadeiros positivos (imagens sem obstáculos), além de apenas 6 falsos positivos e 1 falso negativo.

Figura 13 – Matriz de confusão com os dados de treinamento



Fonte: Os autores (2024).

Esses resultados indicam que, com base nos dados de treinamento, o modelo oferece uma performance consistente e confiável, sendo capaz de equilibrar bem precisão e sensibilidade.

Isso o torna adequado para aplicações onde a detecção precisa de obstáculos é crucial para garantir segurança e eficiência.

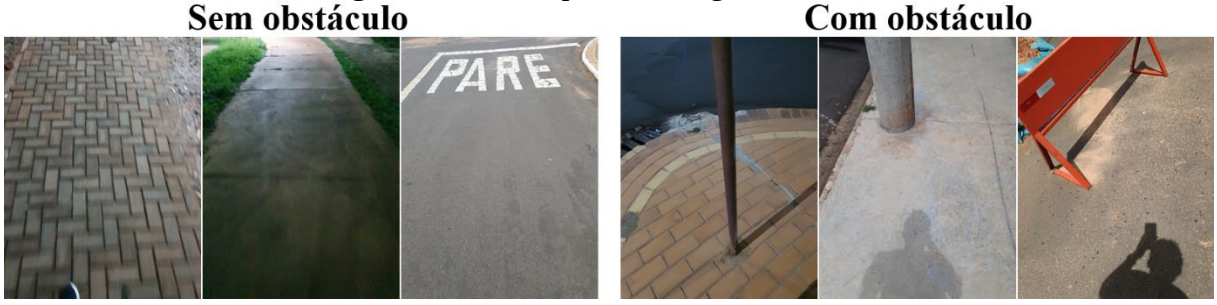
4.2 RESULTADO DOS TESTES REAIS

A partir de um conjunto de imagens capturadas por um usuário enquanto caminhava com o aplicativo, em variados cenários de iluminação e diferentes locais, incluindo diversos tipos de obstáculos e trajetos, será apresentado o desempenho da abordagem descrita na metodologia, assim como possíveis causas de erros e desafios associados ao método.

Para avaliar o desempenho, foram utilizadas 1000 imagens registradas durante o trajeto do usuário. Este conjunto inclui fotos em locais variados, com diferentes níveis de luminosidade e uma variedade de obstáculos. O foco foi dado, preferencialmente, a obstáculos que, embora ausentes na base de treinamento, compartilham características visuais semelhantes, como profundidade e coloração. Esses testes foram conduzidos pelos próprios autores deste estudo, sendo relevante destacar que a abordagem proposta atua como um complemento para o deficiente visual, e não como substituição da bengala, somando-se aos recursos de acessibilidade já disponíveis.

Na Figura 14 são exibidos exemplos das imagens coletadas durante os testes, mostrando cenários com e sem obstáculos, capturados tanto em condições de iluminação natural quanto artificial.

Figura 14 – Exemplos de imagens coletadas

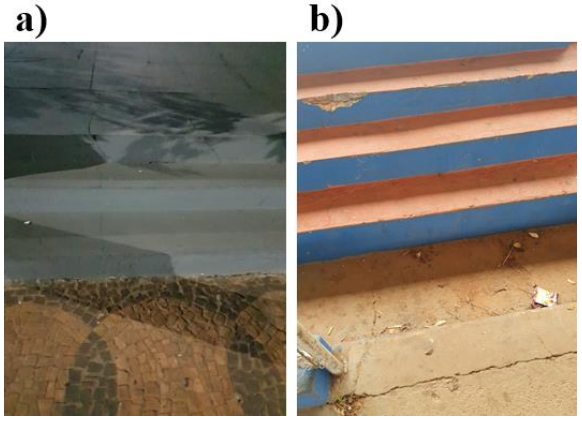


Fonte: Os autores (2024).

Após a coleta e sincronização das imagens, é necessário correlacionar os resultados obtidos com as predições realizadas pelo modelo. Conforme mencionado anteriormente, esse resultado é salvo no título de cada imagem. Então em seguida, foi realizada uma análise manual para classificar cada imagem de acordo com sua situação real, identificando a presença ou ausência de obstáculos.

Durante a etapa de classificação das imagens, observamos que os resultados foram influenciados por uma série de fatores externos. Entre esses, destacam-se a qualidade da imagem capturada pelos dispositivos (como distância, luminosidade e tonalidade de cor entre o chão e o obstáculo), a velocidade de movimento do usuário e a estabilidade do dispositivo em suas mãos. A Figura 15 ilustra esse fenômeno: na imagem (a), uma escada foi fotografada à noite, em condições de baixa luminosidade, levando o modelo a classificá-la incorretamente como livre de obstáculos. Em contrapartida, a imagem (b) apresenta uma escada sob boa iluminação, com um contraste adequado entre os degraus, o que permitiu ao modelo classificá-la corretamente como um obstáculo.

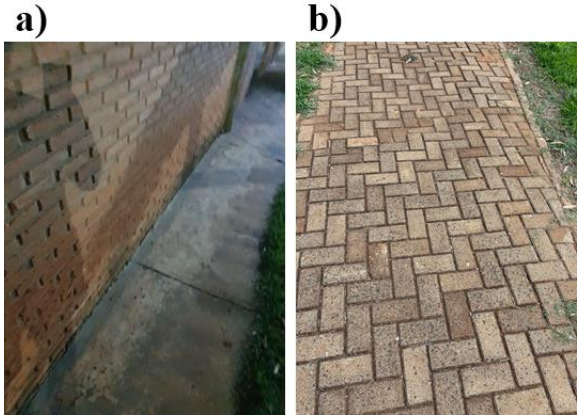
Figura 15 – Exemplos de imagens com diferentes classificações



Fonte: Os autores (2024).

Observamos também algumas situações em que o modelo pode ter generalizado de maneira incorreta. A Figura 16 ilustra esse ponto: na imagem (a), vemos uma parede de blocos que o modelo classificou como livre de obstáculos, enquanto a imagem (b) mostra uma estrada de blocos, sem obstáculos, que foi utilizada durante o treinamento. Como ambas as imagens apresentam um padrão visual semelhante, isso pode ter levado o modelo a realizar uma classificação incorreta.

Figura 16 – Exemplos de imagens com padrões semelhantes



Fonte: Os autores (2024).

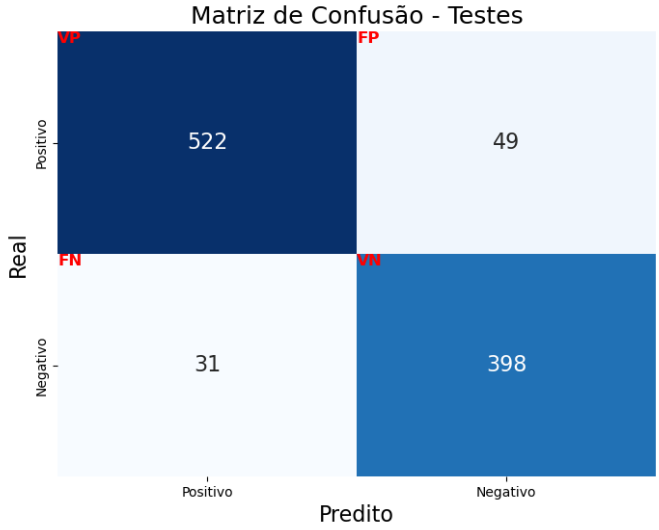
Após as imagens devidamente classificadas, foi realizada a comparação entre a classificação manual e o resultado gerado pela predição do algoritmo.

O modelo alcançou uma precisão de 92,00%, enquanto a pontuação F1 foi de 92,88%, demonstrando um equilíbrio robusto entre a precisão e a sensibilidade do modelo. Com uma sensibilidade (*recall*) de 94,39%, o modelo provou ser muito eficaz em identificar quase todas as áreas livres de obstáculos (verdadeiros positivos). Além disso, uma precisão de 91,42% indica que, entre as previsões de áreas sem obstáculos, a maioria estava correta. A especificidade de 89,04% reforça a capacidade do modelo de identificar corretamente áreas com obstáculos.

A taxa de falsos positivos (FPR) foi de 10,96%, enquanto a de falsos negativos (FNR) foi de 5,61%, ambas indicando uma margem de erro reduzida. O modelo, ao analisar as imagens dos testes reais e do treinamento, apresenta uma leve tendência a cometer mais erros ao prever que uma área está sem obstáculos, pelos motivos já descritos anteriormente.

Esses resultados são demonstrados pela matriz de confusão ilustrada na Figura 17, que mostra 398 verdadeiros negativos (imagens com obstáculos), 522 verdadeiros positivos (imagens sem obstáculos), e apenas 49 falsos positivos e 31 falsos negativos.

Figura 17 – Matriz de confusão com os dados reais



Fonte: Os autores (2024).

Apesar do acurácia menor que o treinamento, provavelmente devido as situações citadas anteriormente, esses resultados mostram que o modelo apresenta um desempenho estável e confiável, conseguindo equilibrar de forma eficaz a precisão e a sensibilidade. Essa característica o torna ideal para aplicações onde a identificação precisa de obstáculos é essencial para garantir tanto a segurança quanto a eficiência. Comprovando os resultados do treinamento.

5 CONSIDERAÇÕES FINAIS

Considerando o presente estudo, foi desenvolvido um modelo para auxiliar deficientes visuais na identificação de obstáculos no caminho, utilizando a técnica de *transfer learning* com a MobileNetV2 para extração de características visuais e um classificador MLP, baseado em redes neurais, para a classificação das imagens. A aplicação proposta usa a câmera do smartphone para capturar imagens, que são processadas localmente, permitindo que o usuário receba alertas sonoros sobre a presença ou ausência de obstáculos.

Este projeto se baseou em um estudo anterior, desenvolvida por Sanga, Polo e Passerini (2023), no qual se utilizava *transfer learning* com a ImageNet (modelos VGG16 e VGG19) e um classificador SVM, resultando em uma acurácia de 75,3%. Com as otimizações realizadas neste trabalho, foi alcançado uma acurácia de 92,00%, representando um aumento significativo de aproximadamente 22,18%. Além disso, diferentemente do projeto anterior, a nova solução permite que todo o processamento ocorra no próprio dispositivo Android, eliminando a necessidade de conexão com um *backend* e proporcionando maior autonomia e rapidez no uso.

Os resultados obtidos em cenários reais demonstraram a efetividade do modelo, mesmo em condições variáveis de iluminação e tipos de obstáculos. Com uma taxa de acertos elevada, o modelo provou ser uma ferramenta promissora para garantir a segurança de pessoas com deficiência visual em seu deslocamento diário, oferecendo uma solução acessível e eficiente, que complementa os recursos tradicionais, como a bengala, para melhorar a independência e a qualidade de vida dos usuários.

REFERÊNCIAS

ACERVO LIMA. **CNN | Introdução à Camada de Pooling**. Disponível em: <https://acervolima.com/cnn-introducao-a-camada-de-pooling/>. Acesso em: 12 maio. 2024.

AWS. **O que é uma rede neural?** Disponível em: <https://aws.amazon.com/pt/what-is/neural-network/>. Acesso em: 11 maio. 2024a.

AWS. **O que é o aprendizado profundo?** Disponível em: <https://aws.amazon.com/pt/what-is/deep-learning/>. Acesso em: 11 maio. 2024b.

BBC NEWS BRASIL. Cegueira afeta 39 milhões de pessoas no mundo; conheça suas principais causas. **BBC**, jun. 2019.

DATA SCIENCE ACADEMY. **Capítulo 8 - Função de Ativação**. Disponível em: <https://www.deeplearningbook.com.br/funcao-de-ativacao/>. Acesso em: 12 maio. 2024.

DIDÁTICA TECH. **Como funcionam as Redes Neurais Convolucionais (CNNs)?** Disponível em: <https://didatica.tech/introducao-a-redes-neurais-convolucionais/>. Acesso em: 12 maio. 2024.

Documentação do Firebase. Disponível em: <https://firebase.google.com/docs?hl=pt-br>. Acesso em: 31 out. 2024.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Cambridge, Mass., USA: MIT Press, 2016.

GOOGLE. **O que é inteligência artificial (IA)?** Disponível em: <https://cloud.google.com/learn/what-is-artificial-intelligence?hl=pt-br>. Acesso em: 11 maio. 2024.

GOOGLE DEEPMIND. **Convolutional Neural Networks for Image Recognition**. Disponível em: <https://storage.googleapis.com/deepmind-media/UCLxDeepMind_2020/L3%20-%20UUCLxDeepMind%20DL2020.pdf>. Acesso em: 12 maio. 2024a.

GOOGLE DEEPMIND. **DeepMind x UCL | deep learning lectures | 3/12 | convolutional neural networks for image recognition**. , jun. 2020b.

HOSPITAL DE OLHOS. **Você sabe o que é acuidade visual?** Disponível em: <https://www.hospitaldeolhos.net/conteudo-blog-hosp/voce-sabe-o-que-e-acuidade-visual/>. Acesso em: 11 maio. 2024.

HUYNH, N. **Convolutional neural networks for image recognition**. Disponível em: <https://medium.com/@nghihuynh_37300/convolutional-neural-networks-for-image-recognition-7148a19f981f>. Acesso em: 12 maio. 2024.

IBGE. **Amostra - Pessoas com deficiência**. Disponível em: <https://cidades.ibge.gov.br/brasil/pesquisa/23/23612?detalhes=true>. Acesso em: 7 maio. 2024.

IBM. **O que são redes neurais convolucionais?** Disponível em: <https://www.ibm.com/br-pt/topics/convolutional-neural-networks>. Acesso em: 12 maio. 2024a.

IBM. **O que é aprendizado por transferência?** Disponível em: <https://www.ibm.com/br-pt/topics/transfer-learning>. Acesso em: 31 out. 2024b.

JAISWAL, S. **Multilayer Perceptrons in Machine Learning: A Comprehensive Guide**. Disponível em: <https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning#rdl>. Acesso em: 31 out. 2024.

Keras. Disponível em: <https://www.tensorflow.org/guide/keras?hl=pt-br>. Acesso em: 31 out. 2024.

ORACLE. **O que é Machine Learning?** Disponível em: <https://www.oracle.com/br/artificial-intelligence/machine-learning/what-is-machine-learning/>. Acesso em: 11 maio. 2024.

SANGA, G. M.; POLO, J. M. G.; PASSERINI, J. A. R. Auxílio a Deficientes Visuais utilizando Redes Neurais Convolucionais In: 5a Semana Integrada de Estudos - FEF, 2023, Fernandópolis SP. **Anais da V Semana Integrada de Estudos - FEF**. Fernandópolis SP: Fundação Educacional de Fernandópolis, 2023, v.1, p.32 – 32. Disponível em: https://revista.fef.br/sie/issue/archive

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **ICLR**, 2015.

SMILKOV, D.; CARTER, S. **Tensorflow - neural network playground**. Disponível em: <http://playground.tensorflow.org>. Acesso em: 31 out. 2024.

SONG, J. et al. A survey of remote sensing image classification based on CNNs. **Big Earth Data**, v. 3, n. 3, p. 232–254, 2019.

TENSORFLOW. **Introdução ao TensorFlow**. Disponível em: <https://www.tensorflow.org/learn?hl=pt-br>. Acesso em: 12 maio. 2024.

UNICESUMAR. **Visão Geral e Desenvolvendo para o Google Android**. Disponível em: <https://sites.google.com/unicesumar.com.br/tec-ads-program-mobile/3%C2%BA-trimestre/li%C3%A7%C3%A3o-19>. Acesso em: 12 maio. 2024.

WORLD HEALTH ORGANIZATION. **Blindness and vision impairment**. Disponível em: <https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment>. Acesso em: 7 maio. 2024a.

WORLD HEALTH ORGANIZATION. **CID-11 para Estatísticas de Mortalidade e de Morbidade**. Disponível em: <https://icd.who.int/browse/2024-01/mms/pt>. Acesso em: 11 maio. 2024b.