RECONHECIMENTO DE SINALIZAÇÃO VIÁRIA ATRAVÉS DE VISÃO COMPUTACIONAL E REDES NEURAIS NO AUXÍLIO DA MOBILIDADE ELÉTRICA EM VEÍCULOS AUTÔNOMOS

Bruno Scremin Brolese [1]

Elton Costa Gomes [2]

João Mota Neto [3]

Marcos Antonio Jeremias Coelho [4]

Resumo: A visão computacional e as redes neurais desempenham um papel crucial na interpretação e processamento dos sinais de trânsito, promovendo uma interação mais inteligente e segura nas estradas. Este estudo buscou implementar o reconhecimento de placas de sinalização viária utilizando um sistema embarcado na plataforma Raspberry PI4 e redes neurais convolucionais em Python. Os resultados demonstraram alta precisão em condições ideais, como boa iluminação e ângulos favoráveis. No entanto, a ampliação da base de dados para incluir uma variedade maior de situações pode potencialmente aumentar a taxa de acertos em cenários diversos, contribuindo para a confiabilidade do método em contextos reais.

Palavras-chave: Visão computacional. Python. Redes Neurais. Sinalização viária.

1 INTRODUÇÃO

O mercado de carros elétricos e a eletrificação dos automóveis têm sido um processo contínuo, ocorrendo há mais de 50 anos e se intensificando na última década. A Plataforma Nacional de Mobilidade Elétrica (PNME) destaca a necessidade crescente de substituir motores de combustão interna por motores elétricos alimentados por baterias recarregáveis, devido à sua eficiência e menor impacto ambiental. Isso está impulsionando investimentos em tecnologias e infraestrutura, afetando a indústria automotiva como um todo. Os conceitos de mobilidade e eletricidade estão intimamente ligados, refletindo uma tendência crescente na indústria automotiva em direção a veículos mais eficientes, sustentáveis e tecnologicamente avançados [1].

[1] Graduado em Engenharia Mecatrônica, 2023. E-mail: scremin.br@gmail.com
[2] Mestrando em Engenharia Metalúrgica, 2024. E-mail: eltoncri@gmail.com
[3] Prof. do Centro Universitário UniSATC. E-mail: joao.neto@satc.edu.br
[4] Prof. do Centro Universitário UniSATC. E-mail: marcos.coelho@satc.edu.br

Com a eletrificação da frota e a evolução dos sistemas eletrônicos de controle, o conceito de carros autônomos foi amplamente implementado. O veículo autônomo é composto por 5 níveis de automação. No Nível 1, o carro possui assistências à direção, como manutenção em faixas, frenagem automática e controlador de velocidade adaptativo. No Nível 2, o carro pode rodar sem intervenção humana por alguns segundos em vias bem sinalizadas. O Nível 3 é similar ao Nível 2, mas sem tempo pré-determinado para rodar. O Nível 4 envolve o carro assumindo total controle da direção, permitindo que o usuário realize outras atividades. Este nível é restrito a rodovias, com o motorista assumindo o controle em áreas urbanas. O último Nível, o 5, representa um veículo 100 % autônomo, capaz de levar o usuário de um ponto a outro sem intervenção humana, superando obstáculos [2].

Dentre os sistemas utilizados em veículos autônomos, a identificação do ambiente ao qual o veículo está imerso é a principal fonte de informação para navegação. Os sistemas de visão computacional se tornaram a principal ferramenta devido ao baixo custo e fácil implementação. Mesmo assim, enfrentam diversos desafios que podem afetar sua eficiência devido às condições climáticas, má condição das sinalizações viárias, vibração do veículo e sombreamento. Métodos como HSI e transformada de Hough são utilizados para compensar esses efeitos [3]. Após localizar a placa, ela pode ser interpretada com precisão por meio de redes neurais, especialmente redes neurais convolucionais (CNN), treinadas para reconhecer padrões e caracteres específicos em sinais de trânsito e ler o texto e os símbolos contidos neles [4].

Este artigo apresenta o desenvolvimento de um sistema de visão computacional para realizar a aquisição de imagens com plataforma embarcada, aplicar topologias para realizar o tratamento de imagens via OpenCV e Python e o treinamento e validação de redes neurais no reconhecimento de sinais de tráfego a ser aplicado em veículos autônomos, a fim de reduzir acidentes e evitar mortes no trânsito. Também é possível auxiliar no desenvolvimento das cidades inteligentes no futuro, criando mais comodidade e segurança para a sociedade.

2 CARROS ELÉTRICOS E AUTÔNOMOS

Após a Segunda Guerra Mundial, a indústria automobilística cresceu significativamente, contribuindo substancialmente para a economia da época e atendendo à crescente necessidade de mobilidade urbana e inter-regional. Essa expansão resultou em uma cadeia produtiva densa e integrada, impulsionada por inovações tecnológicas que melhoraram a segurança, o conforto e a eficiência energética dos veículos. No entanto, a utilização de combustíveis fósseis, previstos para se esgotarem em 2067, e a poluição ambiental associada à queima desses combustíveis destacam a necessidade de transição para veículos elétricos, que utilizam fontes de energia renováveis e têm menor impacto ambiental [5]. Essa mudança requer políticas públicas e instrumentos econômicos e regulatórios para promover a infraestrutura de recarga, o uso do espaço público, a capacitação de recursos humanos e a conscientização sobre os benefícios da mobilidade elétrica nas cidades [1].

O Brasil é um dos maiores produtores e consumidores de veículos no mundo, tendo alcançado a sexta posição em vendas globais de veículos em 2021, com 2.787.850 unidades vendidas, de acordo com a PNME. Houve uma crescente adoção de carros elétricos no mercado brasileiro, com várias empresas incluindo esses veículos em suas frotas. Além disso, o país está investindo em infraestrutura para mobilidade elétrica, como eletropostos e plataformas digitais, e promovendo o desenvolvimento tecnológico por meio de iniciativas como o curso da ANEEL e o programa Rota 2030, visando impulsionar esse mercado [1].

A história dos carros autônomos remonta à década de 1980, quando o professor Ernst Dickmanns iniciou o desenvolvimento desses veículos, utilizando métodos probabilísticos e computação paralela. Na década de 2000, ocorreram avanços significativos nessa área, impulsionados pelos desafios propostos pela agência americana DARPA. Prevê-se que o mercado global de veículos autônomos cresça a uma taxa composta anual de 63,1 % entre 2021 e 2028, de acordo com a *Grand View Research, Inc*. A *National Highway Traffic Safety Administration* (NHTSA) descreve os veículos autônomos em cinco níveis, desde a intervenção total do condutor até a automação completa do veículo [6]. Para alcançar o nível 5 de automação, são necessários sensores como ultrassônicos, GPS, radar e câmeras,

além de tecnologias de visão computacional para perceber o ambiente e tomar decisões adequadas. O desenvolvimento de estradas inteligentes e cidades inteligentes é fundamental para o futuro dos veículos autônomos, permitindo uma conexão eficaz através da Internet das Coisas (IoT) [7]. No entanto, o Brasil enfrenta desafios significativos devido à falta de infraestrutura rodoviária adequada, com apenas 12,3 % da malha rodoviária pavimentada até 2016, segundo a Confederação Nacional de Transportes (CNT). As cidades inteligentes utilizam tecnologia da informação e comunicações para melhorar a gestão e proporcionar uma melhor qualidade de vida aos cidadãos, com características como população, governo, sustentabilidade, mobilidade e qualidade de vida. Curitiba foi considerada a cidade mais inteligente do Brasil em 2023, de acordo com o *Ranking Connected Smart Cities*. O reconhecimento de sinais de tráfego por meio de visão computacional é uma ferramenta crucial para melhorar a eficiência e a segurança do tráfego urbano, permitindo a coleta de dados precisos em tempo real e ajustes na gestão do tráfego para reduzir congestionamentos e melhorar a segurança viária.

2.1 SINALIZAÇÃO VIÁRIA E CIDADES INTELIGENTES

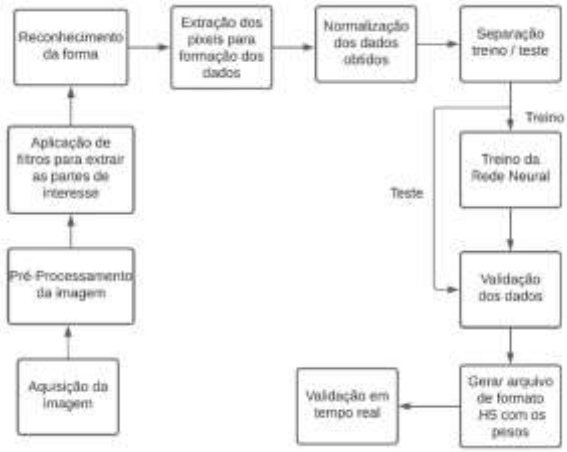
Cidades inteligentes são aquelas que investem em capital humano e social, utilizando Tecnologia da Informação e Comunicações (TIC) para melhorar a gestão e a qualidade de vida dos cidadãos. Essas cidades destacam cinco características principais: população, governo, sustentabilidade, mobilidade e qualidade de vida. Os sinais de tráfego desempenham um papel crucial na gestão do tráfego urbano e na segurança viária. O reconhecimento de sinais de trânsito por meio de visão computacional pode melhorar a eficiência e a segurança do tráfego, permitindo a coleta de dados em tempo real para análise e ajustes na gestão do tráfego, visando reduzir congestionamentos e melhorar a segurança viária [8].

3 PROCEDIMENTO EXPERIMENTAL

O processo de reconhecimento de placas de trânsito utilizando visão computacional e redes neurais é complexo, mas pode ser desdobrado em várias etapas fundamentais, conforme descrito neste estudo. Um fluxograma simplificado

das etapas envolvidas é apresentado na Figura 1. De acordo com este fluxograma, o sistema consiste em duas etapas principais: detecção e reconhecimento das placas de trânsito. As seções subsequentes abordam detalhadamente cada etapa do processo.

Figura 1: Fluxograma do Processo.



Fonte: DO AUTOR

3.1 PROCESSAMENTO DE IMAGEM

O sistema de identificação e classificação de placas de sinalização viária, desenvolvido neste trabalho, foi implementado em Python utilizando o ambiente de desenvolvimento PyCharm, com a maior parte das funções provenientes da biblioteca OpenCV.

Inicialmente, foram coletados dados para o treinamento da rede neural. O processo incluiu a análise manual de placas dentro do campus da UNISATC, mas devido à necessidade de uma quantidade significativa de dados e ao desgaste natural das placas no campus, foram utilizadas também imagens capturadas em diferentes pontos da cidade de Criciúma, SC. Placas de "pare" e "20 km/h" foram capturadas na Rua Porto União, Bairro São Luiz, totalizando cerca de 200 fotos para cada placa. Para aumentar o conjunto de dados, foram adicionadas cerca de 600 imagens de sinalização de trânsito do banco de dados do site CVzone[5].

As imagens foram capturadas em diversos ângulos, iluminações, distâncias e condições de luz para representar situações reais enfrentadas pelo protótipo. Após

[5] https://www.computervision.zone/courses/traffic-sign-classification/

a criação do banco de imagens, foram identificados os pontos de interesse dentro de cada imagem, utilizando duas funções específicas de acordo com a necessidade, sendo a *HoughCircles* e *FindCountours* para detectar círculos e polígonos fechados, respectivamente, nas imagens [9][10].

Em seguida, as imagens foram redimensionadas para 32x32 pixels e aplicados filtros, como a conversão para tons de cinza e um filtro gaussiano para suavizar a imagem e facilitar a detecção de bordas. O detector de bordas Canny foi então aplicado para identificar bordas na imagem.

Finalmente, a detecção de círculos na imagem foi realizada utilizando a função *HoughCircles* para detectar placas que contêm círculos, essencialmente concluindo o processo de identificação e classificação de placas de sinalização viária, como é apresentado na Figura 2.

Figura 2: Detecção pela função HoughCircles.

Fonte: DO AUTOR

A ferramenta *FindContours* do OpenCV, aliada à ferramenta *ContourArea*, foi utilizada para identificar as placas de "PARE", onde identificam contornos fechados na imagem com uma área específica. Além disso, foi implementada uma condição para realizar a detecção de contornos octogonais, visto que a placa de "PARE" é uma figura geométrica com 8 lados iguais, como mostrado na Figura 3.

Figura 3: Detecção pela ferramenta *FindCountours*.

Fonte: DO AUTOR

A avaliação do processamento de imagens foi realizada de forma manual, na qual todas as placas de "pare" e "20 km/h" foram submetidas ao mesmo processo e foi verificado o percentual de detecções positivas e negativas.

3.2 TREINAMENTO, TESTE E GERAÇÃO DO MODELO DA REDE NEURAL CONVOLUCIONAL

O treinamento e os testes da rede neural convolucional foram realizados no ambiente de desenvolvimento PyCharm, usando Python. Foram importadas bibliotecas essenciais, incluindo TensorFlow 2.13.0 e Keras 2.13.1 para o treinamento da rede neural. O Scikit-learn foi utilizado para tarefas de aprendizado de máquina, oferecendo uma ampla gama de algoritmos e métricas. A biblioteca NumPy desempenhou um papel crucial na computação científica. Além disso, o OpenCV e a biblioteca OS foram utilizados para manipulação de imagens e interação com o sistema operacional.

A CNN foi projetada para identificar placas de limite de velocidade de 20 km/h e placas de PARE, organizadas em pastas distintas para suas classes. O conjunto de dados incluiu imagens de treinamento e teste na proporção de 70/30, organizadas em pastas correspondentes a cada classe. A técnica de aumento de

dados foi aplicada usando a função *ImageDataGenerator* do Keras para enriquecer o conjunto de treinamento com variações nas imagens [11].

O treinamento foi realizado em um Notebook Acer Nitro 5, modelo NA-515-45-R91A, com um processador AMD Ryzen 5 da série 5000, 16 GB de memória RAM e placa de vídeo dedicada Nvidia GeForce GTX 1650 com 4 GB de memória de vídeo.

A arquitetura da rede neural inclui camadas de convolução com ativação ReLU, *maxpooling* e *dropout* para evitar o *overfitting*, além de camadas totalmente conectadas. A primeira camada de convolução utiliza 32 filtros com uma convolução de 5x5 pixels para procurar padrões locais nas imagens de entrada, seguida pela aplicação da função de ativação ReLU. É adicionada uma camada de *maxpooling* 2x2 para reduzir a dimensionalidade espacial, seguida por uma segunda camada convolucional com 64 filtros de 3x3 pixels e outra camada de *maxpooling*. Posteriormente, uma camada de *dropout* com taxa de 0,5 é adicionada para prevenir o *overfitting*.

A camada de *flattening* converte a saída anterior em um vetor unidimensional para as camadas totalmente conectadas, chamadas de *Dense*, com 128 neurônios e ativação ReLU. Outra camada *Dense* de saída possui um número de neurônios igual ao número de classes, com função de ativação *Softmax* para gerar probabilidades associadas a cada classe. A função de perda escolhida é a *Categorical_crossentropy*, otimizada pelo algoritmo Adam. Essa arquitetura é projetada para extrair características relevantes das imagens de placas de sinalização, permitindo a classificação precisa das diferentes indicações nelas presentes. A avaliação da rede neural é baseada na precisão, ajustando seus valores de camadas e neurônios para obter resultados que se adéquem ao objetivo da rede.

3.3 APLICAÇÃO EM TEMPO REAL DO PROCESSAMENTO DE IMAGEM E RECONHECIMENTO DE PLACAS

Para a aplicação em tempo real do processamento de imagem e reconhecimento de placas no Raspberry PI 4 com 4GB de RAM, foram utilizadas algumas ferramentas específicas. Inicialmente, é necessário instalar o interpretador Python no Raspberry PI 4. Utilizou-se a versão 3.9.1 do interpretador Python com o compilador PyCharm e as bibliotecas supracitadas para rodar o código de detecção e

carregar o arquivo de formato .h5 da rede neural já treinada. A versão do PyCharm utilizada foi a Community 2023.2.

Após a importação das bibliotecas, é preciso realizar o carregamento do modelo da rede neural com o arquivo de extensão .h5 gerado durante o treinamento. Com o modelo carregado, é necessário iniciar a detecção e tratar cada quadro capturado em tempo real pela câmera Raspberry PI V2 de 8MP, utilizando o mesmo pré-processamento feito no treinamento da rede neural, visto que é preciso que os dados estejam exatamente da mesma forma para se obter um bom resultado de teste e validação.

4 RESULTADOS E DISCUSSÃO

Nesta seção são apresentados os resultados obtidos durante a execução do trabalho, divididos em aquisição e tratamento da imagem, treinamento e testes da rede neural, e aplicação da rede neural em tempo real no laboratório. Os códigos-fonte utilizados neste trabalho estão disponíveis online[6].

4.1 AQUISIÇÃO E TRATAMENTO DA IMAGEM

Nesta fase, foram seguidas algumas etapas para possibilitar a extração das formas da imagem para análise posterior. A primeira etapa foi realizar a detecção dos círculos e dos octógonos que formam as placas de 20 km/h e pare, respectivamente. Para isso, são necessários alguns artifícios, conforme já citado neste trabalho, como os filtros do OpenCV e as funções para extração de círculos e polígonos, respectivamente, para tratar a imagem e assim conseguir extrair a região de interesse e analisar as informações contidas na imagem.

A Figura 4 representa a imagem original sem nenhum filtro. Na Figura 5, é apresentada uma aplicação de conversão em escala de cinza como a primeira etapa do processamento de imagem. Na Figura 6, é aplicado um detector de bordas Canny para evidenciar as bordas e contornos da imagem a fim de identificar regiões de interesse, no nosso caso, as placas de trânsito. Na Figura 7, é aplicado um efeito

[6] https://github.com/RoboticaSATC/Reconhecimento-de-Placas

Gaussiano para gerar um esmaecimento da imagem e assim suavizar as bordas a fim de reduzir ruídos na imagem.

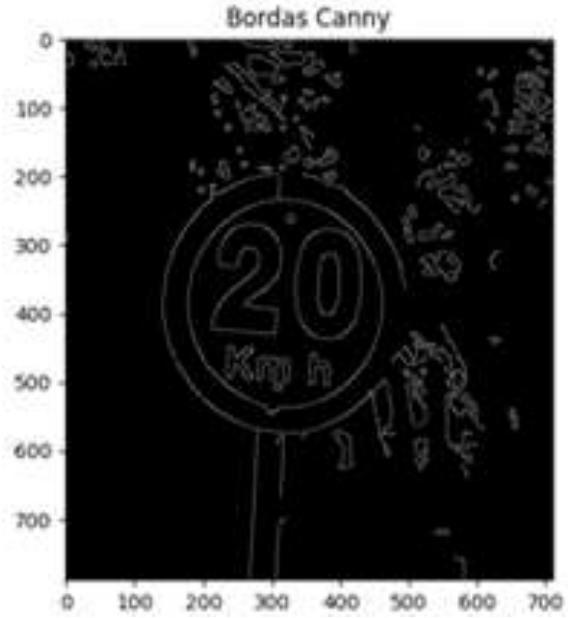
Figura 4: Processamento de imagem: Imagem original.

Fonte: DO AUTOR

Figura 5: Processamento de imagem: Escala de cinza.

Fonte: DO AUTOR

Figura 6: Processamento de imagem: Detector de bordas Canny.



Fonte: DO AUTOR

Figura 7: Processamento de imagem: Efeito Gaussiano.

Fonte: DO AUTOR

Nas Figura 8 e Figura 9, respectivamente, temos as placas selecionadas com as funções HoughCircles e FindContours.

Figura 8: Placa em destaque: 20 km/h.

Fonte: DO AUTOR

Figura 9: Placa em destaque: Pare.

Fonte: DO AUTOR

Como mencionado anteriormente, a avaliação do processamento de imagem para conseguir extrair a região de interesse, onde se encontra a placa, da imagem foi realizada manualmente. Para exemplificar, a Tabela 1 a seguir mostra a eficácia dos filtros aplicados nas imagens para extrair as regiões de interesse. Foram utilizadas 100 imagens de placas de 20 km/h e 100 imagens de placas de PARE para verificar a eficácia da detecção das placas por meio do processamento de imagem.

Tabela 1: Avaliação do processamento de imagens.

HoughCircles		**FindCountours**	
Placa 20 km/h	100 placas	Placa de Pare	100 placas
Acertos	95	Acertos	99
Erros	5	Erros	1

Fonte: DO AUTOR

Pode-se notar que, das 100 imagens de placas de 20 km/h, o processamento conseguiu identificar a existência de uma placa em 95 dos 100 casos testados. Nas imagens das placas de PARE, isso fica ainda mais evidente, com apenas 1 erro dentre as 100 imagens selecionadas para amostragem, mostrando uma grande eficácia em encontrar as placas nas imagens.

4.2 TREINAMENTO E TESTE DO MODELO DE REDE NEURAL CNN

O treinamento da rede neural foi realizado com os parâmetros apresentados na Tabela 2 abaixo.

Tabela 2: Parâmetros de treinamento.

Parâmetros	Valor
Tamanho do lote	30
Etapas por épocas	200
Épocas	50
Dimensões da imagem	(32, 32, 3)

Fonte: DO AUTOR

Foram utilizados vinte por cento das imagens para teste e oitenta por cento para treinamento. Com isso, foram obtidos 315.529 neurônios de entrada para o treinamento da rede neural, oriundos dos dados extraídos das imagens. Os dados apresentados na Tabela 3 representam os valores que a rede neural obteve com as informações de entrada. A primeira camada Conv2D tem uma saída de dimensões 28x28x32, indicando que há 32 filtros aplicados aos dados de entrada, gerando mapas de características 28x28. O valor 832 é o total de parâmetros treináveis nesta camada, que são os pesos.

Após, tem-se uma camada de *Pooling_2D*, onde, após aplicar o *MaxPooling*, as dimensões da entrada são reduzidas pela metade em ambas as dimensões, de 28x28 para 14x14. Logo em seguida, há outra camada *Conv2D* aplicada com 64 filtros, gerando mapas 12x12. Nesta, 18.496 pesos treináveis foram encontrados. Posteriormente, há outra camada de *MaxPooling*, reduzindo ainda mais as dimensões espaciais para 6x6.

Em seguida, há uma camada de *Dropout* para prevenir o excesso de dados. Nesta camada, há 6x6x64 unidades que podem ser temporariamente desativadas durante o treinamento. Após, há uma camada de *Flatten*, transformando os mapas de

características 3D em um vetor unidimensional de tamanho 2.304. Por fim, há duas camadas *Dense* (camadas totalmente conectadas): a primeira com 128 neurônios e 295.040 pesos treináveis, e a outra com 9 neurônios na saída, correspondendo ao número de classes. Nesta camada, foram encontrados 1.161 pesos.

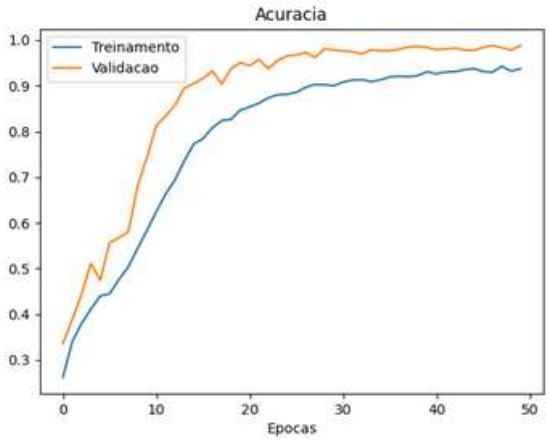
Tabela 3: Parâmetros da Rede Neural.

Camadas	Dados	Parâmetros
Conv2D	(28, 28, 32)	832
Max_Poolling2D	(14, 14, 32)	
Conv2D_1	(12, 12, 64)	18496
Max_Poolling2D_1	(6, 6, 64)	
Dropout	(6, 6, 64)	
Flatten	2304	
Dense	128	295040
Dense_1	9	1161

Fonte: DO AUTOR

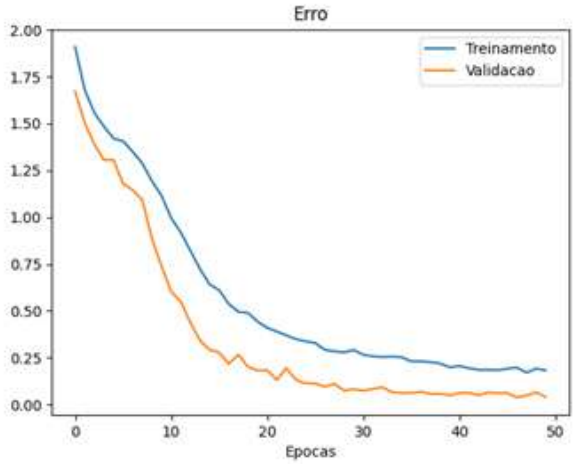
A função de perda escolhida foi a *categorical_crossentropy* e o otimizador escolhido foi o *Adam*, um algoritmo de otimização popular usado no treinamento de redes neurais, especialmente em tarefas de aprendizado profundo. O modelo foi capaz de classificar corretamente mais de 95 % das imagens do conjunto de teste. Os resultados obtidos durante o treinamento e teste da rede neural são representados nas Figura 10 e Figura 11. Pode-se observar que a precisão convergiu para 98 % a partir da trigésima época, obtendo resultados superiores nas épocas posteriores.

Figura 10: Gráfico de métrica da rede neural: Acerto.



Fonte: DO AUTOR

Figura 11: Gráfico de métrica da rede neural: Erro.



Fonte: DO AUTOR

Após, com o código já em funcionamento, ocorre o destaque da região de interesse. Isso acontece quando um círculo ou um contorno de 8 lados é detectado na imagem, representando uma possível placa de trânsito. Esta área é isolada da imagem original e redimensionada para se adequar às exigências do modelo. O modelo treinado é então utilizado para fazer previsões sobre a classe da placa de trânsito na imagem. Este modelo treinado gera um arquivo de formato .h5, utilizado posteriormente na aplicação em tempo real.

4.3 APLICAÇÃO EM TEMPO REAL

A aplicação em tempo real foi realizada primeiramente em notebook e, posteriormente, no Raspberry PI 4 4GB, onde foram obtidos resultados semelhantes em ambos os casos.

Figura 12: Extração da região de interesse em tempo real.

Fonte: DO AUTOR

Como visto, o código extrai a classe prevista e a confiança associada a essa previsão. A função *argmax* determina o índice da classe com a maior

probabilidade, e *amax* obtém o valor máximo de probabilidade. Essas informações são úteis para avaliar a confiança do modelo em suas previsões. Finalmente, os resultados são exibidos no console, fornecendo uma visão rápida da classe identificada e da confiança do modelo. Esse processo é repetido continuamente enquanto o sistema está em execução, permitindo a detecção em tempo real de placas de trânsito e a análise da performance do modelo.

Na detecção em tempo real no Raspberry PI 4, é possível notar uma queda na taxa de *frames* ao realizar a detecção da placa de sinalização. Como mostrado na Figura 13, o FPS antes do reconhecimento da placa está em 25 FPS, e após iniciar o reconhecimento da placa, como mostrado na Figura 14, há uma queda na taxa de quadros, chegando a 2 FPS.

Figura 13: *Frames per second*: Sem reconhecimento de placa de sinalização.

Fonte: DO AUTOR

Figura 14: *Frames per second*: Com reconhecimento de placa de sinalização.

Fonte: DO AUTOR

As Figura 15, 16 e 17 apresentam três imagens destacando diferentes regiões de interesse no processo de reconhecimento de sinalização viária. Cada figura representa uma forma geométrica específica detectada pelo sistema, demonstrando a eficácia do modelo na identificação de sinais de trânsito.

Na Figura 15, temos uma placa circular que foi identificada e destacada corretamente pelo sistema. A previsão correta indica que o modelo conseguiu classificar com precisão o tipo de sinalização representada pela forma circular.

Figura 15: Regiões de interesse: Circular destacada e analisada com a previsão correta.

Fonte: DO AUTOR

A Figura 16 mostra uma placa octogonal, comumente associada à sinalização de "PARE". O sistema conseguiu detectar e destacar a placa octogonal, e a previsão foi correta, confirmando a robustez do modelo na classificação de formas geométricas específicas.

Figura 16: Regiões de interesse: Octogonal destacada e analisada com a previsão correta.

Fonte: DO AUTOR

Na Figura 17, vemos novamente uma placa circular destacada. Embora a subfigura não mencione explicitamente a previsão, a identificação correta da região de interesse sugere que o sistema está funcionando conforme o esperado para este tipo de sinalização.

Figura 17: Regiões de interesse: Circular destacada.

Fonte: DO AUTOR

Essas imagens exemplificam o processo de detecção e classificação de sinais de trânsito, demonstrando como o modelo pode isolar e analisar regiões de interesse de diferentes formatos geométricos, contribuindo para a segurança viária por meio da automação e reconhecimento de sinais de trânsito.

5 CONCLUSÃO

Neste estudo, foi desenvolvida uma ferramenta para detecção de sinais de trânsito utilizando redes neurais em uma plataforma embarcada. A introdução do reconhecimento de sinalização viária por meio de visão computacional e redes neurais representa um avanço significativo na mobilidade elétrica em veículos elétricos e autônomos, contribuindo para um ambiente mais seguro nas estradas.

Para garantir resultados satisfatórios da rede neural, além de uma arquitetura adequada, é essencial que o processo de treinamento e validação seja realizado com qualidade.

A rede neural utilizada demonstrou alta precisão quando testada em condições de laboratório, alcançando taxas de acerto de até 100 % em alguns casos com boa iluminação e imagens legíveis. Foi observado que uma quantidade menor de camadas na rede neural não apresentou desempenho satisfatório, enquanto um modelo com 7 camadas obteve resultados excelentes neste estudo.

Para trabalhos futuros, uma ideia seria capturar as fotos utilizando a câmera específica do Raspberry PI 4 e dedicar mais tempo à etapa de aquisição de dados, criando manualmente um banco de dados maior de imagens para incluir mais placas em condições reais. Além disso, seria interessante explorar outros modelos de redes neurais para verificar se é possível melhorar ainda mais os resultados obtidos.

REFERÊNCIAS

[1] 2° ANUÁRIO BRASILEIRO DE MOBILIDADE ELÉTRICA. Brasília: **PNME**, v. 2, 2023. Acessado em 15/03/2023.

[2] ANDERSON, James M. et al. **Autonomous Vehicle Technology**. Santa Monica, Calif: Rand, 2016.

[3] SILVA, José Demisio Simões da. **USO DE REDES NEURAIS EM VISÃO COMPUTACIONAL E PROCESSAMENTO DE IMAGENS**. 2004. 70 f. Tese

(Doutorado) - Curso de Estágio Spe/Rhae, Nuclear Engineering Department da Universidade Of Tennessee, Knovville, 2004.

[4] OLIVEIRA, A. F., SANTOS, L. G. (2017). **Traffic sign detection and recognition using deep learning**: a review. Machine Learning and Applications, 4, 1-13.

[5] STOPFER, Nicole; SOARES, Anuska; CASTRO, Nivalde José de; ROSENTAL, Rubens. **A mobilidade elétrica na américa Latina**. Rio de Janeiro: E-Papers, 2021. 284 p.

[6] PFLEGER, Sergio Genilson. **Sistema de navegação robótica por visão computacional**. 2013. 83 f. Tese (Doutorado) - Curso de Ciências da Computação, Universidade Federal de Santa Catarina, Florianópolis, 2013.

[7] PINDARWATI, **SMART Highway**: Near Future Highway Systems in Jakarta. The Third Asia Future Conference, 2016.

[8] GAMA-CASTRO, J et al. **A traffic sign detection system for smart cities**. IEEE International Conference on Smart Cities, Green Computing and Communications (SmartCity), 2014.

[9] ANTONELLO, Ricardo. **Introdução a Visão Computacional com Python e OpenCV**. Luzerna: [S.N.], 2017. Disponível em: https://professor.luzerna.ifc.edu.br/ricardo- antonello/wp-content/uploads/sites/8/2017/02/Livro-Introdu%C3%A7%C3%A3o-a- Vis%C3%A3o-Computacional-com-Python-e-OpenCV-3.pdf. Acesso em: 06 dez. 2023.

[10] ESTEVAM, Rodrigo. **HoughCircles** - Detecção de círculos em imagens com OpenCV ePython. 2021. Disponível em: https://medium.com/turing-talks/houghcircles- detec%C3%A7%C3%A3o-de-c%C3%ADrculos-em-imagens-com-opencv-e-python- 2d229ad9d43b. Acesso em: 06 dez. 2023.

[11] LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. **Deep learning**. Nature, [S.L.], v. 521, n. 7553, p. 436-444, 27 maio 2015.

ABSTRACT

Computer vision and neural networks play a crucial role in interpreting and processing traffic signs, promoting a smarter and safer interaction on the roads. This study aimed to implement the recognition of road signs using an embedded system on the Raspberry PI4 platform and convolutional neural networks in Python. The results demonstrated high accuracy under ideal conditions, such as good lighting and favorable angles. However, expanding the database to include a greater variety of situations may potentially increase the accuracy rate in diverse scenarios, contributing to the reliability of the method in real-world contexts.

Key-words: Computer vision; Python; Neural Networks; Road signs.

AGRADECIMENTOS

O presente trabalho foi realizado com apoio da Fundação de Amparo à Pesquisa e Inovação de Santa Catarina (FAPESC) - Código de Financiamento 2022TR002172