Universidade Federal da Paraíba
Centro de Informática
Programa de Pós-Graduação em Informática

Dissertação de Mestrado

ARQUITETURAS DE REDES NEURAIS PARA CONDUÇÃO DE VEÍCULOS AUTÔNOMOS TERRESTRES EM ESTRADAS BRASILEIRAS SIMULADAS

Victor Miranda Fernandes

João Pessoa - PB
Fevereiro - 2020

Universidade Federal da Paraíba
Centro de Informática
Programa de Pós-Graduação em Informática

ARQUITETURAS DE REDES NEURAIS PARA CONDUÇÃO DE VEÍCULOS AUTÔNOMOS TERRESTRES EM ESTRADAS BRASILEIRAS SIMULADAS

Victor Miranda Fernandes

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Informática da Universidade Federal da Paraíba como parte dos requisitos necessários para obtenção do grau de Mestre em Informática.

Área de Concentração: Computação Distribuída

Profa. Dra. Thaís Gaudencio do Rêgo
Orientadora

João Pessoa - PB
Fevereiro - 2020

Catalogação na publicação
Seção de Catalogação e Classificação

F363a Fernandes, Victor Miranda.
Arquiteturas de Redes Neurais para Condução de Veículos Autônomos Terrestres em Estradas Brasileiras Simuladas / Victor Miranda Fernandes. - João Pessoa, 2020.
69 f.

Orientação: Thaís Gaudencio do Rêgo.
Dissertação (Mestrado) - UFPB/Informática.

1. Veículos autônomos. 2. Deep Learning. 3. CNN. 4. Processamento de Imagem. I. Rêgo, Thaís Gaudencio do. II. Título.

UFPB/BC

UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Ata da Sessão Pública de Defesa de Dissertação de Mestrado de Victor Miranda Fernandes, candidato ao título de Mestre em Informática na Área de Sistemas de Computação, realizada em 20 de fevereiro de 2020.

Aos vinte dias do mês de fevereiro, do ano de dois mil e vinte, às dez horas, no Centro de Informática da Universidade Federal da Paraíba, em Mangabeira, reuniram-se os membros da Banca Examinadora constituída para julgar o Trabalho Final do Sr. Victor Miranda Fernandes, vinculado a esta Universidade sob a matrícula nº 20181000644, candidato ao grau de Mestre em Informática, na área de "Sistemas de Computação", na linha de pesquisa "Computação Distribuída", do Programa de Pós-Graduação em Informática, da Universidade Federal da Paraíba. A comissão examinadora foi composta pelos professores: Thais Gaudêncio do Rego (PPGI-UFPB), Orientadora e Presidente da Banca, Tiago Maritan Ugulino de Araujo (PPGI-UFPB), Examinador Interno, Yuri de Almeida Malheiros Barbosa (UFPB), Examinador Externo ao Programa, Carlos Danilo Miranda Regis (IFPB), Externo à Instituição. Dando início aos trabalhos, a Presidente da Banca cumprimentou os presentes, comunicou aos mesmos a finalidade da reunião e passou a palavra ao candidato para que o mesmo fizesse a exposição oral do trabalho de dissertação intitulado: "Uma Arquitetura de Rede Neural para Condução de Veículos Autônomos Terrestres em Estradas Brasileiras". Concluída a exposição, o candidato foi arguido pela Banca Examinadora que emitiu o seguinte parecer: "***aprovado***". Do ocorrido, eu, Ruy Alberto Pisani Altafim, Coordenador do Programa de Pós-Graduação em Informática, lavrei a presente ata que vai assinada por mim e pelos membros da banca examinadora. João Pessoa, 20 de fevereiro de 2020.

Prof. Dr. Ruy Alberto Pisani Altafim

Profª. Thais Gaudêncio do Rego
Orientador (PPGI-UFPB)

Thaís Gaudencio do Rêgo

Prof. Tiago Maritan Ugulino de Araujo
Examinador Interno (PPGI-UFPB)

Tiago Maritan U. de Araújo

Prof. Yuri de Almeida Malheiros Barbosa
Examinador Externo ao Programa (UFPB)

Yuri de Almeida Malheiros Barbosa

Prof. Carlos Danilo Miranda Regis
Examinador Externo ao Programa (IFPB)

Carlos Danilo M. Regis

RESUMO

Os veículos autônomos são aqueles que funcionam sem um condutor humano, sendo um dos produtos que mais recebe investimento da indústria automobilística no seu desenvolvimento, com mais de 50 bilhões de dólares investidos nos últimos 5 anos. Uma das formas de implementá-los é utilizando técnicas de inteligência artificial e, apesar de já existirem diversas implementações, poucos estudos focam na condução em estradas com problemas de infraestrutura, como por exemplo: buracos, lama, estradas de barro, etc, que podem resultar em acidentes de trânsito e danos aos veículos. Isso se torna relevante, levando em consideração dados sobre rodovias brasileiras, onde, atualmente, apenas 13,7% delas são pavimentadas. Neste sentido, uma arquitetura de rede neural convolucional para conduzir veículos autônomos em estradas degradadas é proposta neste trabalho. A arquitetura foi treinada com dados extraídos do jogo *Euro Truck Simulator 2* com um mapa brasileiro realístico que considera os problemas de infraestrutura citados anteriormente. A modelagem e treinamento da arquitetura foi realizada em duas fases: a primeira utilizando dados coletados via teclado e a segunda utilizando dados coletados a partir de um volante joystick e com a adição de técnicas de processamento de imagens. Os resultados mostraram que a arquitetura treinada na segunda fase obteve um desempenho superior em tempo médio para intervenção humana, ou seja, o tempo para o condutor intervir na arquitetura treinada pelos dados do volante foi cerca de 9 vezes maior quando comparado ao tempo para a arquitetura treinada com os dados coletados via teclado.

Palavras-chave: Veículos autônomos, Deep Learning, CNN, Processamento de Imagem.

ABSTRACT

Autonomous vehicles are those that operate without a human driver, being one of the products that receive more investment from the automobile industry in its development, with more than 50 billion dollars invested in the last 5 years. One of the ways to implement them is using artificial intelligence techniques and, although there are already several implementations, few studies focus on driving on roads with infrastructure problems, such as: potholes, mud, etc., which can result in traffic accidents and vehicle damage. This becomes relevant, taking into account data on Brazilian highways, where, currently, only 13.7 % of them are paved. In this sense, a convolutional neural network architecture to drive autonomous vehicles on degraded roads is proposed in this work. The architecture was trained with data extracted from the game *Euro Truck Simulator 2* with a realistic Brazilian map that considers the infrastructure problems mentioned above. The modeling and training of architecture was carried out in two phases: the first using data collected via the keyboard and the second using data collected from a joystick steering wheel and with the addition of image processing techniques. The results showed that the architecture trained in the second phase obtained a superior performance in average time for human intervention, that is, the time for the driver to intervene in the architecture trained by the steering wheel data was about 9 times greater when compared to the time for the architecture trained with the data collected via the keyboard.

Keywords: Autonomous Vehicles, Deep Learning, CNN, Image Processing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABS	*Anti-lock Braking System* (Sistema de Travagem Antibloqueio)
ACC	*Autonomous Cruise Control* (Controle Automático de Navegação)
CNN	*Convolutional Neural Network* (Redes Neurais Convolucionais)
DNIT	Departamento Nacional de Infraestrutura e Transportes
ELU	*Exponential Linear Unit* (Unidade Linear Exponencial)
ESP	*Electronic Stability Program* (Programa de Estabilidade Eletrônica)
IA	Inteligência Artificial
ISO	*International Organization for Standardization* (Organização Internacional para Padronização)
LIDAR	*Light Detection And Ranging* (Detecção de Luz e Alcance)
LReLU	*Leaky Rectified Linear Unit* (Unidade Linear Retificada)
LSTM	*Long Short-Term Memory* (Memória de curto prazo)
MLNN	*Multi-Layer Neural Network* (Rede neural de múltiplas camadas)
PReLU	*Parametric Rectified Linear Unit* (Unidade Linear Retificada Paramétrica)
ReLU	*Rectified Linear Unit* (Unidade Linear Retificada)
SAE	*Society of Automotive Engineers* (Sociedade de Engenheiros Automotivos)

SUMÁRIO

1 INTRODUÇÃO

Um veículo autônomo é aquele com direção e velocidade controlados por um agente inteligente que deve continuamente tomar decisões sobre como se comportar no trânsito de maneira consistente com as regras e restrições com as quais foi programado [1]. A Sociedade de Engenheiros Automotivos, do inglês *Society of Automotive Engineers* (SAE), definiu 6 níveis de automação para veículos autônomos (0 a 5) [2]. Os níveis de 0 a 3 requerem um motorista humano habilitado, já os níveis 4 e 5 permitem que o carro seja conduzido de forma automática, sem a necessidade de um motorista humano habilitado. A seguir temos o detalhamento de cada nível de autonomia.

- Nível 0 - Sem automação

  O condutor humano tem total controle do veículo em 100% do tempo. Este nível pode conter componentes que estabilizam o trajeto do veículo como ABS (Sistema de Travagem Antibloqueio, do inglês *Anti-lock Braking System*) que permite freios eficazes sem travamento da roda, ou ESP (Programa de Estabilidade Eletrônica, do inglês *Electronic Stability Program*), que reconduz o automóvel para a trajetória original em caso de desestabilização. Carros comuns no mercado brasileiro fazem parte deste nível.

- Nível 1 - Direção assistida

  O condutor humano e o sistema compartilham o controle do veículo, como por exemplo o sistema ACC (Controle Automático de Navegação, do inglês *Autonomous Cruise Control*) que controla a aceleração do veículo enquanto o condutor humano controla a direção. Este nível pode conter assistência de estacionamento, onde o estacionamento paralelo (baliza), perpendicular ou angular são realizados de maneira automática. O motorista deve estar pronto para tomar o controle do veículo a qualquer momento.

- Nível 2 - Automação parcial

  O sistema tem total controle de aceleração, frenagem e direção do veículo, porém, o condutor humano deve monitorar a condução automática e estar preparado para intervir imediatamente caso o sistema não funcione de maneira adequada. Alguns modelos de automóveis necessitam que o condutor mantenha as mãos no volante enquanto o sistema conduz o veículo. Um exemplo deste nível seria um veículo que consegue se manter alinhado à faixa.

- Nível 3 - Automação condicional

  A partir deste nível, o sistema é capaz de monitorar totalmente o ambiente externo. O condutor humano pode desviar a atenção do trajeto, como por exemplo enviar uma mensagem no celular ou assistir um filme. Ainda é esperado que o condutor humano esteja pronto para interceder a qualquer alerta emitido pelo sistema e retomar o controle do automóvel. É comum que exista um limite máximo de velocidade de até 60km/h neste nível.

- Nível 4 - Alta automação

  O sistema é capaz de conduzir o veículo mesmo nos casos em que o condutor humano não responda às intervenções. A condução automática só é suportada em áreas limitadas ou em condições especiais como engarrafamentos. Fora dessas circunstâncias, o sistema deve ser capaz de abortar o trajeto e estacionar o veículo caso o condutor humano não retome o controle. Este nível permite que o condutor humano durma ou se retire do assento de motorista enquanto o sistema opera o veículo.

- Nível 5 - Automação total

  Nenhuma intervenção humana é necessária neste nível. O motorista se tornaria apenas um passageiro e o veículo não precisaria ter direção nem pedais. Um exemplo seria um táxi-robô.

Uma série de projetos-piloto de carros autônomos foram criados nos últimos anos. Um fato comum entre esses projetos-piloto é que parte das tarefas de direção, como a percepção ambiental, o planejamento do caminho ou mesmo o controle do volante, são realizados por abordagens baseadas em aprendizagem profunda (do inglês, *Deep Learning*). Com a demonstração bem-sucedida dos protótipos autônomos de aprendizagem profunda, o foco da indústria automotiva está gradualmente passando da construção e da demonstração de protótipos de veículos para a produção em série. Agora, o grande desafio passa a ser como colocar as redes neurais em carros de produção em série, em conformidade com a segurança [3].

Recentemente, o *Deep Learning* demonstrou grande sucesso na aprendizagem supervisionada em diversas áreas como: reconhecimento de imagem, reconhecimento de fala, análise de sentimento e navegação por robôs [4]. No entanto, esses avanços não foram levados em conta pela Organização Internacional para Padronização (ISO, do inglês *International Organization for Standardization*), que propôs em 2011 o padrão ISO 26262, responsável por regular a segurança em veículos rodoviários [5]. O padrão inclui requisitos e recomendações para todo o ciclo de vida da fabricação de automóveis, desde a fase de conceito até a operação e o serviço. O principal objetivo da ISO 26262 foi ajudar a

indústria automotiva a abordar questões de segurança funcional de uma maneira mais sistemática. O padrão foi definido sem levar em consideração o aprendizado profundo, dado que a primeira versão da ISO 26262 foi publicada antes da popularização do *Deep Learning* [3].

Os primeiros modelos de aprendizagem profunda para veículos autônomos urbanos foram baseados nos sensores LIDARs (Detecção de Luz e Alcance, do inglês *Light Detection And Ranging*), que geralmente custam dezenas de milhares de dólares. O alto custo limita suas aplicações em larga escala aos veículos comuns. Recentemente, mais atenção é dada aos métodos baseados em imagem, nos quais o LIDAR é substituído por uma ou mais câmeras barateando o custo para aplicações em larga escala [6].

Aplicações baseadas em aprendizagem profunda requerem uma quantidade grande e variada de dados para permitir que a rede neural consiga boa generalização. Uma vez que existe muita variação nas cenas de tráfego no mundo real, é muito difícil assegurar uma cobertura de 100% das cenas do mundo real através de um único conjunto de dados, independentemente do seu tamanho. Além disso, a maioria dos bancos de dados contêm apenas cenas de tráfego normal que contribuiriam muito pouco para atualizar os pesos sinápticos [1] durante o processo de ajuste fino, dado um modelo que já é pré-treinado. São as anomalias no conjunto de dados, como acidentes de trânsito, infraestrutura precária, etc., que são de maior interesse, mas difíceis de obter [3].

Com base nessas informações e levando em conta a condição precária das estradas brasileiras, com a maior parte das rodovias não pavimentadas (como detalhado na Seção 1.1), é interessante que o funcionamento dos veículos autônomos seja avaliado nesse tipo de ambiente. Sendo assim, neste trabalho são propostas arquiteturas de redes neurais treinadas a partir de imagens e dados gerados no simulador *Euro Truck Simulator 2* utilizando cidades brasileiras simuladas com tráfego de veículos e rodovias realistas. As arquiteturas são testadas e comparadas entre si e seus resultados são apresentados.

1.1 MOTIVAÇÃO

O Brasil é um país de dimensões continentais onde optou-se por investir majoritariamente em rodovias em vez de ferrovias. De acordo com um estudo publicado em 2014 pelo Ilos [7], 67% da carga transportada pelo Brasil em 2012 foi movimentada por rodovias, enquanto apenas 18% por ferrovias. O estudo do Ilos ainda mostra que, para transportar mil toneladas de carga em uma ferrovia brasileira, é preciso gastar R$ 43 por quilômetro. Já nas rodovias esse valor é de R$ 259 (seis vezes maior). O alto custo de manutenção das

[1] Pesos sinápticos são usados para armazenar o conhecimento das redes neurais durante o processo de treinamento.

rodovias Brasileiras acarreta nas condições precárias que são encontrados alguns trechos rodoviários .

De acordo com o Índice de Viabilidade de Veículos Autônomos (do inglês, *Autonomous Vehicles Readiness Index*), realizado pela *KPMG International* [8], 20 países foram escolhidos de acordo com sua economia e foram avaliados em 4 pilares: Política e legislação, Tecnologia e inovação, Infraestrutura e Aceitação do consumidor. O Brasil ficou entre os 4 piores países (17ª colocação) sendo o último colocado em Política e legislação e penúltimo colocado em infraestrutura.

A colocação do Brasil no estudo da *KPMG International* pode ser corroborada pelo Anuário Estatístico de Transportes 2010 - 2016 [9], em que é informado que o Brasil possui 1.435.855 Km de rodovias, sendo apenas 196.601 Km pavimentados, cerca de 13,7%.

Outro órgão que reflete a condição do Brasil com relação à infraestrutura das rodovias é o Departamento Nacional de Infraestrutura e Transportes (DNIT). Este órgão possui um serviço de consulta de condições das rodovidas disponível em `http://servicos.dnit.gov.br/condicoes`, no qual as rodovias são separadas por trechos referente aos estados. Cada trecho é separado em subtrechos que são classificados como *Boa Viagem*, *Atenção*, *Cuidado* ou *Sem Informação*. A partir de um levantamento nos dados fornecidos pelo serviço de consulta do DNIT, foi possível observar que dos 285 trechos, 87 (mais de 30%) não possuíam informação alguma, e dentre os trechos que possuem informação, 39% apresentam ao menos 1 subtrecho em condição de *Atenção* ou *Cuidado*. A má qualidade das rodovias Brasileiras pode impactar diretamente no funcionamento de carros autônomos em nosso país. Dessa forma, neste trabalho, modelos de redes neurais especializadas em conduzir veículos em estradas com estado de conservação degradado foram desenvolvidas e o seus resultados são apresentados.

1.2 OBJETIVOS

1.2.1 OBJETIVO GERAL

Conduzir de forma autônoma o veículo do *Euro Truck Simulator 2* em estradas com estado de conservação degradado e com fluxo de veículos seguindo a rota dada pelo GPS.

1.2.2 OBJETIVOS ESPECÍFICOS

- Desenvolver um script coletor de dados para o jogo *Euro Truck Simulator 2*;

- Construção de base de dados com dados de navegação em estradas com estado de conservação degradado;
- Implementar arquiteturas de redes neurais para processar os dados obtidos.
- Treinar as arquiteturas implementadas com os dados coletados.
- Validar e comparar as arquiteturas treinadas com missões do simulador escolhido.

1.3 ESTRUTURA DA DISSERTAÇÃO

No Capítulo 2 é apresentada a fundamentação teórica sobre redes neurais convolucionais (CNN, do inglês *Convolutional Neural Networks*), que é a base para o desenvolvimento deste projeto. No Capítulo 3 são descritos os trabalhos relacionados. No Capítulo 4 são apresentadas as metodologias utilizadas tanto para a coleta de dados, quanto para a arquitetura escolhida. No Capítulo 5 são discutidos os resultados obtidos. E por fim, no Capítulo 6, são apresentadas as considerações finais.

2 REDES NEURAIS CONVOLUCIONAIS

2.1 INTRODUÇÃO

A habilidade das Redes Neurais de Múltiplas Camadas (MLNN, do inglês *Multi-Layer Neural Network*), treinadas com gradiente descendente, de aprender relações complexas, multi-dimensionais e não lineares a partir de uma grande coleção de exemplos as tornam candidatas óbvias para tarefas de reconhecimento de imagem [10]. No entanto, com o aumento de resolução das câmeras atuais, as arquiteturas MLNN se tornam inviáveis pela grande quantidade de parâmetros treináveis e, consequentemente, grande quantidade de memória necessária para o treinamento da rede neural. Além disso, as características locais da imagem são perdidas já que a imagem precisa ser aplainada para servir como entrada para a rede MLNN, tornando a rede neural incapaz de classificar corretamente imagens com deslocamento ou escala diferentes.

A fim de resolver estes problemas, [10] propõe as redes neurais convolucionais, que combinam três ideias arquiteturais para garantir algum grau de invariância de deslocamento, escala e distorção. São elas: campos receptivos locais, pesos compartilhados e subamostragem espacial.

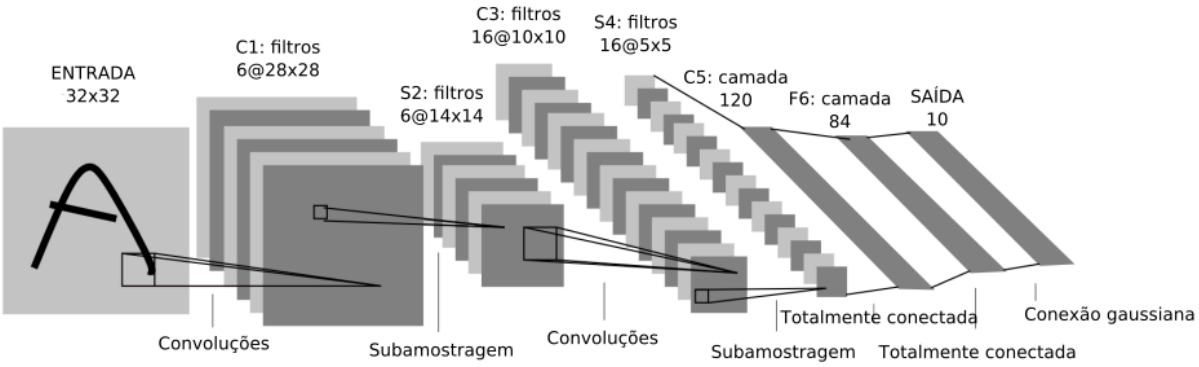
2.2 ARQUITETURA

A arquitetura base de uma Rede Neural Convolucional (CNN) é mostrada na Figura 2.1, em que é possível observar camadas convolucionais e camadas de subamostragem intercaladas que são responsáveis pela extração de características da imagem de entrada. Por fim, tem-se as camadas totalmente conectadas que processam as características extraídas e as mapeiam para a saída da rede. Esta arquitetura contém as três características propostas por [10] que serão detalhadas a seguir.

2.2.1 CAMPOS RECEPTIVOS LOCAIS

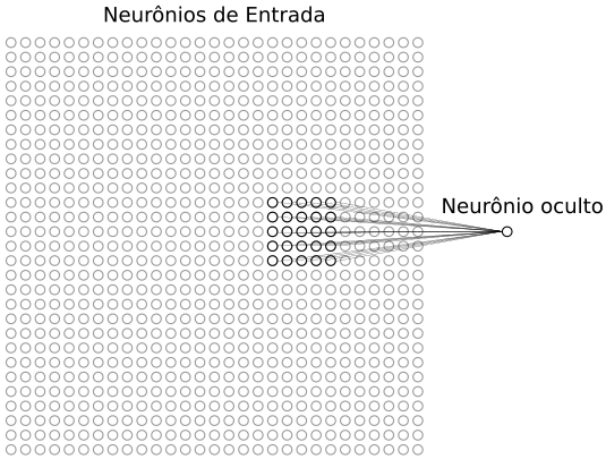
De acordo com [11], a ideia dos campos receptivos locais é manter a estrutura espacial de pequenas janelas na imagem de entrada. Como nas redes MLNN, os pixels da imagem de entrada serão conectados à primeira camada oculta, porém, nem todos os pixels serão conectados a todos os neurônios. Na verdade, cada neurônio da primeira camada oculta será conectado a uma pequena região da imagem de entrada, por exemplo uma janela de dimensões 5 x 5 pixels como está mostrado na Figura 2.2, correspondendo a 25 pixels de entrada.

Figura 2.1 – Arquitetura de Rede Neural Convolucional criada por [10] para classificação de caracteres manuscritos, chamada de LeNet-5.



Cada conexão representa um peso (parâmetro treinável), e cada neurônio oculto também possui um viés (*bias*) tornando o neurônio capaz de aprender a analisar seu campo receptivo local [11].

Figura 2.2 – Região da imagem de entrada chamada de campo receptivo local.



O campo receptivo local é deslocado de acordo com um valor de *stride*, que é o passo de deslocamento em pixels para formar a camada seguinte. Nas Figuras 2.3 e 2.4 é mostrado um deslocamento horizontal para a direita de passo 1.

É possível observar nas Figuras 2.3 e 2.4, que a primeira camada oculta tem dimensões menores do que a camada de entrada. Isso se deve ao modo como o deslocamento do campo receptivo local foi feito. Existem dois modos possíveis, o primeiro é chamado de *valid padding*, onde o campo receptivo local começa sobreposto ao primeiro conjunto de pixels válido da imagem de entrada, como é o caso da Figura 2.3. Assumindo que o campo receptivo local tem dimensões K x K pixels e a imagem de entrada tem dimensões I x I pixels, a resolução da camada seguinte será dada por:

$$I - (K - 1) \text{ x } I - (K - 1) \text{ pixels} \tag{2.1}$$

Figura 2.3 – Campo receptivo local do primeiro neurônio da camada oculta.

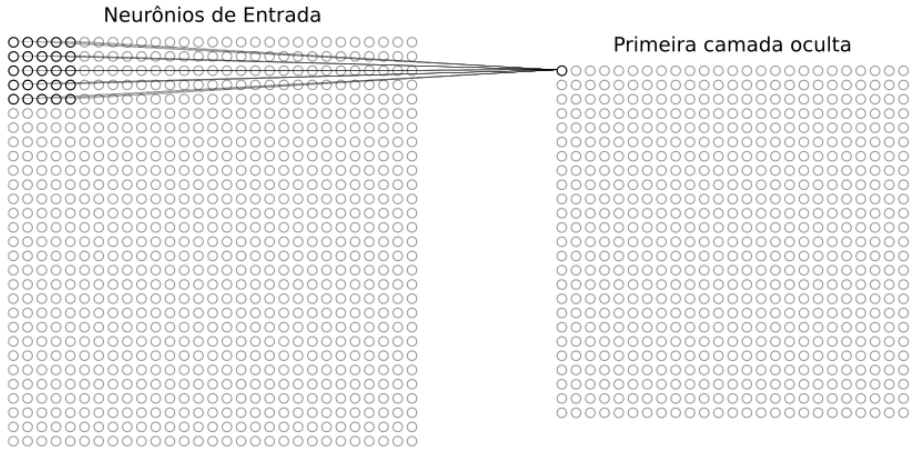
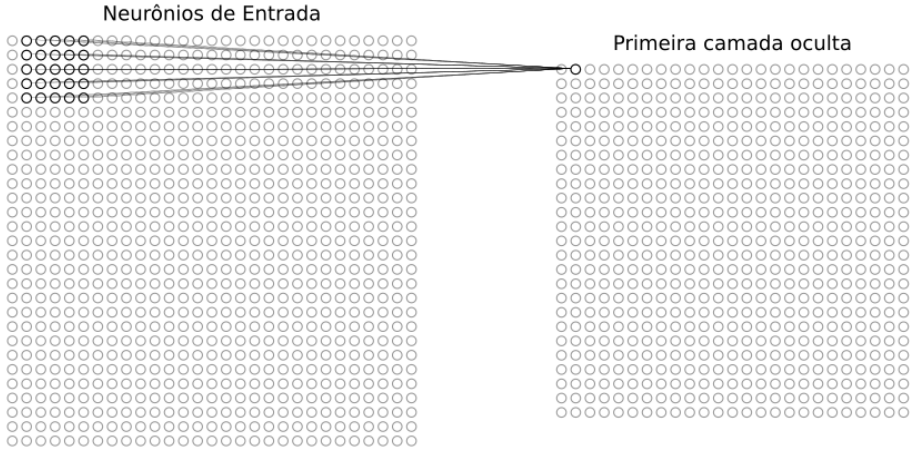


Figura 2.4 – Campo receptivo local do segundo neurônio da camada oculta.



Ou seja, se a imagem de entrada tem dimensões 32 x 32 pixels ($I = 32$) e o tamanho do campo receptivo local for 5 x 5 pixels ($K = 5$), a resolução da primeira camada oculta será:

$$32 - (5 - 1) \text{ x } 32 - (5 - 1) = 28 \text{ x } 28 \text{ pixels} \tag{2.2}$$

O segundo modo é chamado de *same padding* ou *zero padding*, onde uma borda de zeros com largura igual a $(K - 1)/2$ é adicionada na imagem, fazendo com que a próxima camada tenha as mesmas dimensões da camada atual. Para demonstrar este fato, seja uma imagem de entrada com dimensões 32 x 32 pixels e o campo receptivo local com dimensões 5 x 5 pixels, será adicionado uma borda de largura igual a $(5-1)/2 = 2$ à imagem de entrada, aumentando suas dimensões para 36 x 36 pixels. Utilizando a Equação 2.1 para estes valores, as dimensões da primeira camada oculta serão $(36 - (5 - 1))$ x $(36 - (5 - 1)) = 32$ x 32 pixels, que é exatamente o tamanho original da imagem de entrada.

2.2.2 PESOS COMPARTILHADOS

Como dito anteriormente, cada neurônio da primeira camada oculta, no exemplo da Figura 2.2, possui um *bias* e 5 x 5 pesos conectados ao seu campo receptivo local. Entretanto, todos os 28 x 28 neurônios da camada oculta compartilham os mesmos valores de pesos e *bias*. Dessa forma, a saída para o k-ésimo neurônio será dada por:

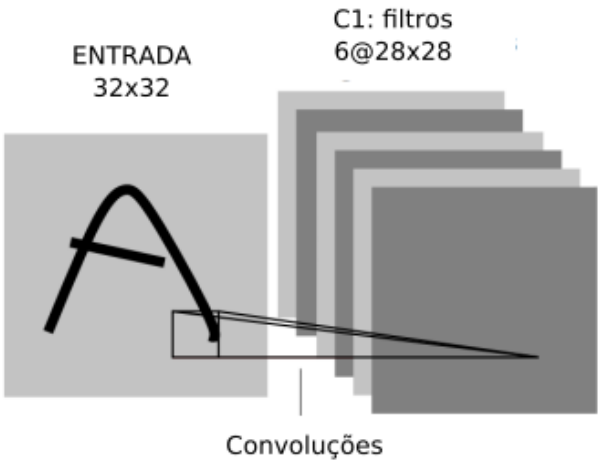
$$\sigma(b + \sum_{l=0}^{4} \sum_{m=0}^{4} a_{j+l,k+m} w_{l,m}) \tag{2.3}$$

Onde σ é a função de ativação do neurônio, b é o *bias*, $a_{x,y}$ é o valor do pixel da imagem de entrada na posição x, y e $w_{l,m}$ é o array 5 x 5 com os pesos.

Isto significa que todos os neurônios da primeira camada detectam exatamente a mesma característica, mas em locais diferentes da imagem [11]. Uma característica poderia ser, por exemplo, uma borda ou um padrão geométrico mais abstrato. O conjunto de pesos compartilhados treinados para reconhecer uma característica específica é chamado de filtro ou *kernel*.

Usualmente, cada camada da rede convolucional pode detectar várias características diferentes simultaneamente, ou seja, cada camada possui vários filtros. Então, se cada filtro é uma matriz bidimensional, cada camada com vários filtros se torna uma matriz tridimensional. Este fato pode ser observado na Figura 2.5, onde é mostrada a primeira camada convolucional da LeNet-5 da Figura 2.1. Nesta camada existem 6 filtros de dimensões 28 x 28.

Figura 2.5 – Primeira camada convolucional da LeNet-5. Imagem adaptada de [10].



A grande vantagem de utilizar camadas convolucionais é a pequena quantidade de parâmetros treináveis [11]. A extração de 6 características efetuada na Figura 2.5, utiliza

6 filtros de dimensões 5 x 5, totalizando $6 * 5 * 5 = 150$ parâmetros. Já em uma rede MLNN, a primeira camada densa para extração de 6 características teria $32 * 32 * 6 = 6144$ parâmetros.

Os padrões das características tornam-se mais complexos e diversificados com o aumento da profundidade da rede [12]. Na Figura 2.6 pode-se observar que a primeira camada detecta padrões simples que se assemelham a texturas. A segunda camada já consegue detectar bordas verticais, horizontais e diagonais. As camadas mais profundas detectam padrões tão complexos que é praticamente impossível de saber o que esses padrões representam da imagem de entrada.

Figura 2.6 – Cada linha mostra filtros de uma mesma camada, onde as camadas são de cima para baixo. Imagem retirada de [12].

2.2.3 SUBAMOSTRAGEM ESPACIAL

Normalmente, após cada camada convolucional, existe uma camada de subamostragem (*pooling layer*) [11]. O papel desta camada é reduzir a quantidade de dados que servirá como entrada para a próxima camada a fim de reduzir a quantidade de parâmetros treináveis e tornar o treinamento da rede neural mais eficiente.

O tipo mais comum de *pooling* é o *max-pooling* que é exemplificado nas Figuras 2.7 e 2.8. Assim como as camadas convolucionais, a camada de *max-pooling* possui dois parâmetros, o tamanho da janela e o *stride*, que define o passo do deslocamento da janela. Normalmente se utiliza janelas de dimensão 2 x 2 pixels e *stride* de 2, para não haver sobreposição entre as janelas como é mostrado na Figura 2.7.

No *max-pooling*, o maior valor encontrado em cada janela é passado para a saída. Com janelas de dimensão 2 x 2 pixels e *stride* de 2, a camada de *max-pooling* diminui cada dimensão pela metade, tornando a imagem 4 vezes menor como está mostrado na Figura 2.8.

Figura 2.7 – Exemplo matemático da operação *max-pooling*. Imagem retirada de https://computersciencewiki.org.

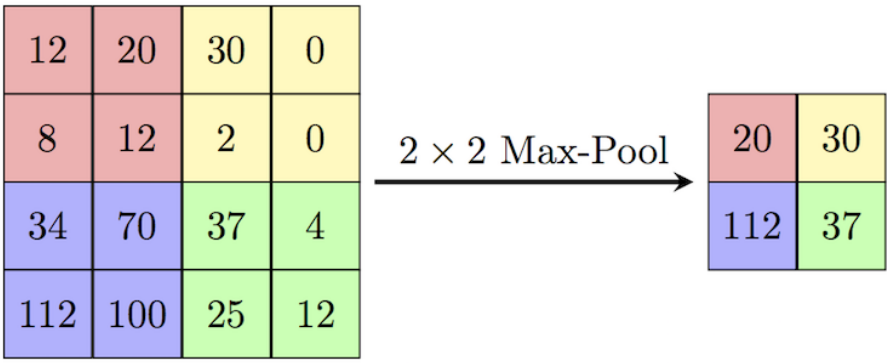
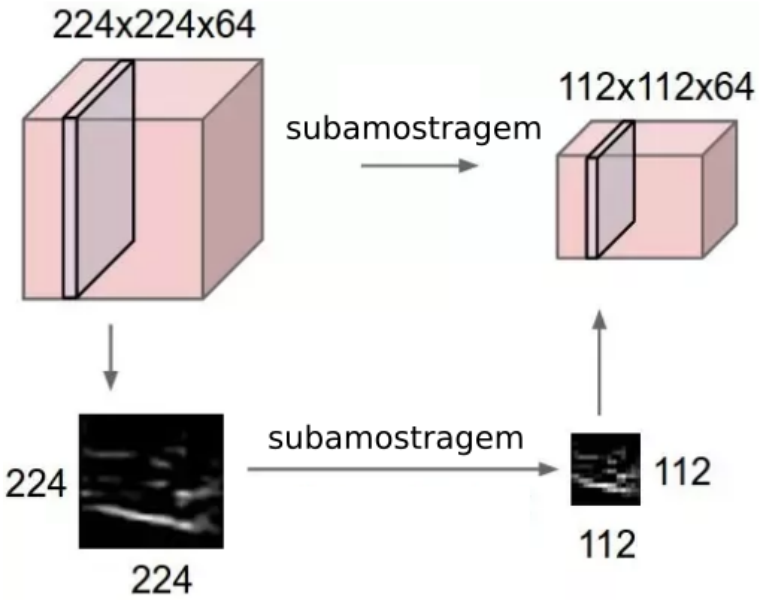


Figura 2.8 – Exemplo real da operação *max-pooling*. Imagem adaptada de https://computersciencewiki.org.



Existem também os tipos *min-pooling* e *average-pooling*. No *min-pooling*, o menor valor encontrado na janela é passado para saída. Já no *average-pooling*, a média dos valores da janela que é passado para a saída.

Entre os três tipos, o *max-pooling* é o mais utilizado por ser computacionalmente mais eficiente do que o *average-pooling* e por representar melhor a informação da imagem do que o *min-pooling*.

2.2.4 OUTROS ASPECTOS DA ARQUITETURA

Após a extração das características da imagem de entrada feita por uma sequência de camadas convolucionais e camadas de subamostragem, os filtros da última camada convolucional são aplainados e passados por uma ou duas camadas densas. Nos problemas de classificação, a última camada densa possui uma quantidade de neurônios definida pela quantidade de classes que se deseja classificar. O papel dessas camadas densas é identificar quais características encontradas fazem parte de quais classes e prosseguir com a classificação. Esta habilidade de identificação dos padrões nas características extraídas, bem como a própria extração das características são treinadas com base em exemplos (conhecimento prévio) e este processo de aprendizagem será descrito no tópico a seguir.

2.3 PROCESSO DE APRENDIZAGEM

2.3.1 INTRODUÇÃO

O processo de aprendizagem de redes CNN se assemelha ao processo nas redes MLNN. O dado de entrada (uma imagem, uma matriz no caso das CNN) é propagado pelas camadas da rede até chegar na última camada. A última camada resulta na classificação da imagem de entrada e possibilita que um valor de perda (ou erro) seja calculado, já que, durante o treinamento supervisionado, a cada imagem de entrada inserida na rede, sabe-se a qual classe a imagem pertence. Uma vez que o valor de perda foi calculado, este é retropropagado pela saída da rede até chegar nas entradas com a finalidade de identificar quais neurônios contribuíram com o valor da perda para terem seus pesos atualizados de modo a minimizar o valor da perda.

A retroprogação do erro da saída até chegar às entradas pode ser feito pelos modos estocástico, lote e mini-lote. No modo estocástico, a cada imagem inserida, um valor de perda é calculado e retropopagado na rede para que os pesos sejam atualizados.

No modo lote, todas as imagens da base de treino são passadas para a rede e um valor médio de perda é calculado e retropropagado na rede para que os pesos sejam atualizados. Neste modo a rede tem maior poder de generalização e o treinamento acontece de forma mais rápida se comparado com o modo estocástico, porém, uma grande quantidade de memória é necessária para armazenar a propagação de todas as imagens da base de treino, sendo assim, nem sempre é possível efetuar o treinamento neste modo.

O modo mini-lote foi criado como um meio termo entre o modo estocástico e o modo lote. Neste modo, a cada grupo de imagens que são passadas para a rede, um valor de perda médio do grupo é calculado e retropropagado na rede para que os pesos sejam atualizados. O mini-lote é o modo mais utilizado atualmente por ter um maior poder de generalização do que o estocástico e por requerer menos memória do que o modo lote.

Nas seções que seguem serão apresentadas todas as etapas do treinamento da rede CNN.

2.3.2 ALIMENTAÇÃO DA REDE (*FEEDFORWARD*)

Como mencionado na Equação 2.3, cada camada possui sua função de ativação. A saída da função de ativação de uma camada serve como entrada para a próxima camada. As funções de ativação têm como objetivo adicionar não linearidades no processo de aprendizagem para permitir que a rede neural consiga solucionar problemas complexos.

Para problemas de processamento de imagens, existem algumas funções de ativação que facilitam a aprendizagem da rede e se tornaram um padrão entre os arquitetos de redes CNN. As principais funções de ativação para as camadas convolucionais das redes CNN estão mostradas na Figura 2.9 e serão detalhadas a seguir.

- Unidade Linear Retificada (ReLU, do inglês *Rectified Linear Unit*)

  Esta função de ativação foi pensada inicialmente para resolver o problema de *vanish gradient*, que acontece quando redes neurais com várias camadas têm seu gradiente tendendo a zero, interrompendo o treinamento da rede. Esse problema acontecia com as principais funções de ativação utilizadas para redes neurais, como por exemplo: sigmoidal e tangente hiperbólica.

  A função matemática da ReLU pode ser vista na Equação 2.4. Esta função possui $gradiente = 1$ para todos os valores positivos de x, evitando o problema de *vanish gradient*.

$$f(x) = \begin{cases} x & \text{se } x > 0 \\ 0 & \text{caso contrário} \end{cases} \tag{2.4}$$

- Unidade Linear Exponencial (ELU, do inglês *Exponential Linear Unit*)

  Variação da função ReLU onde a parte negativa é exponencial. A função matemática da ELU pode ser vista na Equação 2.5. Existe um fator a que serve como um limite

mínimo para a função. No caso da ELU da Figura 2.9, temos $a = 1$.

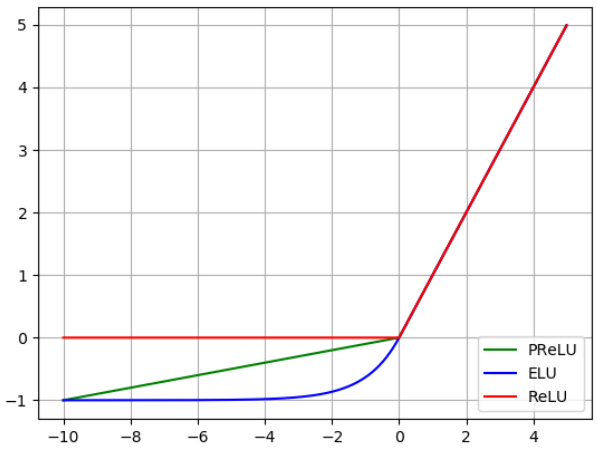
$$f(x) = \begin{cases} x & \text{se } x > 0 \\ a(e^x - 1) & \text{caso contrário} \end{cases} \tag{2.5}$$

- Unidade Linear Retificada Paramétrica (PReLU, do inglês *Parametric Rectified Linear Unit*)

  Variação da função ReLU onde é permitido um gradiente positivo para valores negativos de x. A função matemática da PReLU pode ser vista na Equação 2.6. O gradiente para valores negativos de x pode ser definido diretamente pelo fator a. Quando $a = 0,01$, a PReLU é chamada de LReLU (Unidade Linear Retificada com Vazamento, do inglês *Leaky Rectified Linear Unit*). No caso da PReLU da Figura 2.9, temos $a = 0,1$, um valor baixo, mas não baixo o suficiente para se tornar uma LReLU.

$$f(x) = \begin{cases} x & \text{se } x > 0 \\ ax & \text{caso contrário} \end{cases} \tag{2.6}$$

Figura 2.9 – Comparação entre as funções de ativação ReLU, ELU e PReLU.



As funções de ativação citadas acima são utilizadas tanto nas camadas convolucionais, quanto nas camadas densas. Porém, os neurônios da última camada da rede CNN utilizam uma função de ativação diferente. A função de ativação mais utilizada para problemas de classificação é a *Softmax*, que possui uma saída semelhante a uma distribuição de probabilidades, pois a soma de todas as saídas é unitária. Esta propriedade da função *Softmax* que gera uma distribuição de probabilidade a torna adequada para interpretação probabilística em tarefas de classificação.

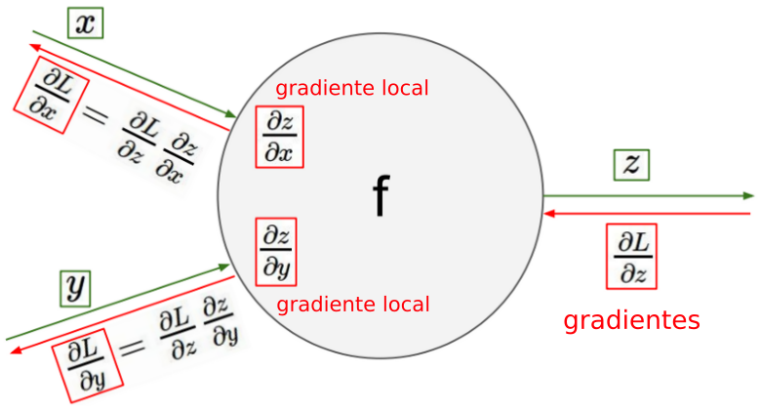
A função matemática da *Softmax*, também conhecida como função exponencial normalizada, (do inglês *Normalized Exponential Function*), pode ser vista na Equação 2.7, onde é recebido um vetor como entrada (a) e gerado um vetor normalizado de saída. Este vetor de saída (p) é resultado da exponencial de cada elemento dividido pela somatória da exponencial de todos os elementos do vetor de entrada, como é mostrado na Equação 2.7. Cada elemento da saída representa uma classe na qual se deseja classificar a imagem. A saída de maior valor (maior probabilidade) define a qual classe a imagem de entrada pertence.

$$p_i = \frac{e^{a_i}}{\sum_{k=1}^{N} e_k^a} \tag{2.7}$$

2.3.3 APRENDIZAGEM POR RETROPROPAGAÇÃO (*BACKPROPAGATION*)

A ideia por trás da aprendizagem por retropropagação (*backpropagation*) é avaliar o quanto cada neurônio contribuiu com o valor do erro para que estes tenham seus pesos atualizados, minimizando a função perda. Esta minimização é feita movendo o erro no sentido contrário ao gradiente atual da função perda, o chamado gradiente descendente (*gradient descent*). Neste processo, o gradiente da função perda em relação a saída da rede é inserido de volta na rede pela saída e propagado até a camada de entrada como mostrado na Figura 2.10. A cada camada, os gradientes locais são calculados pela regra da cadeia e utilizados na atualização dos pesos da camada atual.

Figura 2.10 – Exemplificação da regra da cadeia. Adaptado de https://becominghuman.ai.



A função de ativação que define a saída de um neurônio pode ser vista como uma função que recebe duas variáveis como entrada: os pesos e a saída da camada anterior. Seja f uma função de ativação de um neurônio, x a saída da camada anterior, y os pesos da camada atual, z a saída do neurônio e L a função perda, é possível observar na Figura 2.10 que o gradiente da perda pode ser utilizado para calcular os gradientes em relação aos

pesos e em relação a saída da camanda anterior. Com o gradiente em relação aos pesos, pode-se atualizá-los de forma a minimizar o valor da função perda. O gradiente em relação a saída da camada anterior pode ser propagado para a camada anterior para que seus pesos também sejam atualizados de forma a minimizar a função perda.

3 TRABALHOS RELACIONADOS

Em um estudo recente [3] foram elencados três desafios para a *Deep Learning* aplicada em veículos autônomos. São eles: Completude do banco de dados, Implementação da Rede Neural e Transferência de aprendizado. Os três desafios foram tomados como ponto de partida nesta pesquisa e o detalhamento do estado da arte em cada um dos desafios será dado a seguir.

3.1 COMPLETUDE DO BANCO DE DADOS

A completude do banco de dados é uma característica importante, já que a abordagem mais utilizada para detecção do ambiente ainda é a aprendizagem supervisionada [3]. Um modelo supervisionado só é capaz de aprender o que sua base de treino oferece como conhecimento, então, quanto mais próximo do real for o ambiente contido no banco de dados, mais conhecimento é adquirido pela rede neural, tornando-a mais assertiva.

Os veículos autônomos comerciais da atualidade ainda estão no nível SAE 3, ou seja, só são capazes de se autoconduzir em condições específicas. Tais condições não incluem falhas de infraestrutura que são comumente encontradas nas rodovidas e estradas de países de terceiro mundo. A fim de prototipar um veículo autônomo capaz de conduzir nesses países, é necessário que um banco de dados com imagens do tráfego e dados de navegação destes países seja utilizado como base para o treinamento dos algoritmos. Um levantamento em bancos de dados com informações reais de câmeras e sensores foi feito por [13], no qual 36 bancos de dados foram encontrados e analisados. A maioria deles contêm dados de países de primeiro mundo focados em: alterações climáticas, detecção de pedestres, tráfego em rodovias, etc. Apenas dois bancos de dados contêm informações de países latino americanos, porém, ambos apresentam alguns pontos negativos que inviabilizam suas utilizações neste trabalho. São eles:

- *CCSAD Dataset* [14] - Dados coletados apenas no México, com imagens em escala de cinza, dificultando a detecção de sinais e placas. Existe a dificuldade de testar um algoritmo treinado nesta base, já que necessita-se de um veículo real em uma estrada real.
- *Mapillary Vistas Dataset* [15] - Contém imagens de ruas e avenidas de vários países, incluindo o Brasil. Porém, não fornece dados de navegação, apenas dados para segmentação e classificação das imagens.

Assim, criar um banco de dados diversificado e numeroso para o desenvolvimento de carros autônomos é indispensável. Dessa forma, a etapa de coleta de dados é uma etapa necessária para os pesquisadores dessa área. Em [16] é proposto um novo conjunto de dados, Chinese Driving from a Bike View, adquirido através de uma câmera de alta resolução acoplada a uma bicicleta, a fim de obter imagens de ciclovias e diversificar os bancos de dados existentes, pensando não apenas em em carros autônomos, mas também em cadeiras de rodas, bicicletas e alguns robôs. No entanto, este trabalho, apesar de apontar a diversidade como sua principal contribuição, tem o custo da câmera e dos circuitos associados.

Em [17], os autores dividem o carro autônomo em três camadas. Uma camada de aquisição de dados, uma camada de processamento de dados e uma camada de atuação. A camada de aquisição de dados envolve muitos componentes de hardware, como sensores infravermelho, radares de curto e longo alcance, detecção de luz e alcance (LIDAR), câmeras e dispositivos de comunicação (transceptores). Os dados coletados são normalmente processados pelo computador de bordo do carro (camada de processamento) e depois são utilizados pela camada de atuação para acionar o controle de manobras, freios, entre outros. A confiabilidade dos dados coletados é um aspecto muito importante, que força o uso de componentes de hardware mais precisos e, portanto, mais caros.

Uma maneira de reduzir o custo do coletor de dados para fins de pesquisa é usar carros de controle remoto (RC) com sensores e câmeras menores, o que torna o experimento mais barato, como foi feito em [18]. O DeepPicar é uma replicação em pequena escala de um carro autônomo real chamado DAVE-2 pela NVIDIA [19]. O hardware usado por esta réplica consiste em um Raspberry Pi 3 Modelo B de quatro núcleos, uma câmera e um pequeno carro de controle remoto. O valor estimado seria de cerca de $ 100 no total. No entanto, apesar de mais baratos, os dados coletados não são tão confiáveis quando comparados com ambientes virtuais realistas que apresentam condições de tráfego, infraestrutura complexa de rodovias, como ocorre em alguns jogos e simuladores. O uso de ambientes virtuais também permite redução de custos na implementação de carros autônomos.

Nos trabalhos [20] e [21], o simulador desenvolvido pela Udacity é usado no desenvolvimento de arquiteturas de redes neurais para controlar um carro autônomo, mas esse simulador não tem condições de tráfego nem representa um ambiente urbano. Isso leva a uma diminuição na diversidade de dados coletados para o treinamento de redes neurais.

Na falta de um banco de dados que represente a realidade brasileira e tendo em vista que alguns estudos, já relatados anteriormente, apontam que a utilização de ambientes virtualizados facilita não só a coleta de dados, mas que também se torna mais barata e

mais rápida, como também o teste dos algoritmos [22] [13], neste trabalho optou-se por utilizar um ambiente virtual tanto para etapa de coleta de dados, como para etapa de testes.

Em [13] é fornecida uma visão geral sobre os ambientes virtuais para testes de veículos autônomos, na qual foram analisados 21 ambientes virtuais. No entanto, apenas 4 são compatíveis com a linguagem Python, que é a mais utilizada para projetos de *Machine Learning* de acordo com [23]. Todos os 4 ambientes compatíveis com Python apresentam pontos negativos que dificultam suas utilizações neste trabalho, são eles:

- *AirSim (Microsoft)* - requer habilidades em modelagem 3D para criação de uma cidade que reflita a realidade brasileira, com falhas em infraestrutura.
- *CARLA* - também requer habilidades em modelagem 3D para criação de uma cidade que reflita a realidade brasileira, com falhas em infraestrutura.
- *OpenDaVINCI & OpenDLV* - o programa para manipulação do mapa não foi encontrado na URL disponibilizada pelo *GitHub* do projeto.
- *V-Rep (Coppelia)* - gráficos em baixa qualidade e requer habilidades em modelagem 3D para criação de uma cidade que reflita a realidade brasileira, com falhas em infraestrutura.

Na falta de um simulador que seja de fácil adequação à realidade da infraestrutura brasileira, [22] propõe a utilização de jogos realistas como ambiente virtual para testes de carros autônomos. Porém, a maioria dos jogos possuem cidades pré-estabelecidas e estas, normalmente, são baseadas nos países da Europa e América do Norte. Sendo assim, necessitariam da criação de MODs (abreviação de *modifications*, que são qualquer tipo de modificações e customizações do jogo, seja em mapas, fases, personagens, design, etc) para que estas cidades fossem adequadas ao interesse desta pesquisa.

A criação de MODs não é uma tarefa trivial, já que envolve desenvolvimento de jogos e modelagem 3D. No entanto, o jogo *Euro Truck Simulator 2* permite a utilização de diversos tipos de MODs. Dentre os MODs disponíveis, um deles é o Mapa Eldorado [24], que representa algumas cidades do estado de São Paulo de maneira fidedigna como mostrado nas Figuras 3.1, 3.2 e 3.3, o que torna este jogo uma possibilidade para ser utilizado nesta pesquisa.

Figura 3.1 – Estrada de barro com buracos no Mapa Eldorado.

Figura 3.2 – Estrada de asfalto com buracos no Mapa Eldorado.

Figura 3.3 – Área urbana com lombada no Mapa Eldorado.

3.2 IMPLEMENTAÇÃO DA REDE NEURAL

O primeiro problema com relação à implementação da rede neural é a definição do modo de aprendizado a ser utilizado. Majoritariamente, os trabalhos recentes em *Deep Learning* aplicado a veículos autônomos são baseados em duas abordagens de treinamento: aprendizado supervisionado e aprendizado por reforço. Diversos trabalhos utilizando as duas abordagens constam na literatura apresentando bons resultados [25] [26] [27] [28].

A abordagem de aprendizado por reforço não precisa de um banco de dados, pois o treinamento se dá pelo *feedback* do ambiente onde o agente inteligente está inserido. Ações indesejadas recebem uma recompensa negativa e as ações desejadas recebem uma recompensa positiva. A ideia é que o agente, ao longo do treinamento, consiga ser otimizado para receber o máximo de recompensas positivas e o mínimo de recompensas negativas.

A abordagem de aprendizado supervisionado aplicada em veículos autônomos é comumente chamada de aprendizado por imitação ou clonagem comportamental. Pois, um motorista especialista deve conduzir um veículo a fim de coletar dados comportamentais (ações do motorista, como: ângulo da direção, aceleração do veículo, velocidade do veículo, etc.) e ambientais (imagens ao redor do veículo e dados de sensores) de modo que a rede neural possa identificar e aprender quais padrões nas imagens e sensores levaram o condutor a realizar suas ações. Neste tipo de treinamento é esperado que a rede neural consiga conduzir veículos tão bem quanto o condutor especialista, pelo menos em situações que estão representadas no banco de dados.

Uma comparação entre os métodos supervisionado e por reforço aplicados em veículos autônomos utilizando o mesmo ambiente virtual foi feita por [25]. No entanto, como o intuito do autor era de comparar apenas arquiteturas de aprendizado por reforço entre si, seu trabalho se mostra enviesado já que sua implementação baseada no método supervisionado poderia ser melhorada em alguns aspectos (como: regularização, variação nos hiperparâmetros da rede neural, melhorar a completude do banco de dados, etc.) como o próprio autor menciona em suas conclusões.

Além disso, o método por reforço requer que o ambiente, caso seja simulado, entregue as recompensas para o agente inteligente de forma automática. Como optou-se por utilizar um jogo comercial (de código fechado) como ambiente virtual, essa automatização do ambiente fica impossibilitada, pois não é possível modificar seu código fonte. A única forma de contornar esta situação é criando uma interface para que um especialista retorne as recompensas para o agente inteligente manualmente. Assim sendo, e levando em conta que outros trabalhos citados anteriormente conseguiram resultados expressivos utilizando a abordagem de aprendizado supervisionado, esta foi a abordagem escolhida para implementação da rede neural utilizada neste trabalho.

3.3 TRANSFERÊNCIA DE APRENDIZADO

A transferência de aprendizado consiste na utilização de uma rede neural já treinada como base para um treinamento adicional. Este procedimento é feito para reduzir o tempo de treinamento das redes, uma vez que já são iniciadas com algum conhecimento. Na área de processamento de imagens, uma rede neural já treinada é capaz de identificar algumas características da imagem, como por exemplo: bordas, formas geométricas, cores, etc. Em [29] é utilizado uma VGG net [30] pré-treinada no *ImageNet Dataset*, que, de acordo com [31], é um banco de dados com mais de 465 mil imagens separadas em 200 classes.

Neste trabalho optou-se por não utilizar transferência de aprendizado, tendo em vista que a proposta é criar e utilizar uma base de dados referente ao tráfego no Brasil, algo pioneiro de acordo com a revisão do estado da arte realizado nesta pesquisa. Então, é esperado que o tempo gasto não utilizando uma rede pré-treinada seja compensado pelo aprendizado focado apenas nas características importantes a respeito do tráfego no Brasil.

3.4 CONTRIBUIÇÕES DO TRABALHO

Neste tópico, as principais contribuições científicas referentes a este trabalho são apresentadas. O desenvolvimento deste trabalho traz contribuições envolvendo dois grandes aspectos dentro do contexto de inteligência artificial: o banco de dados e a arquitetura da rede neural desenvolvida. Cada um desses aspectos serão analisados nos tópicos que seguem.

3.4.1 BANCO DE DADOS

De acordo com a pesquisa bibliográfica realizada neste trabalho, não foi encontrado nenhum trabalho que utiliza um banco de dados com estradas brasileiras, levando em conta seus problemas de infraestrutura como rachaduras, buracos, entre outros. Sendo assim, este seria um trabalho pioneiro sobre veículos autônomos utilizando inteligência artificial que o treinamento da rede neural é baseado na real infraestrutura das cidades brasileiras. Assim, a primeira contribuição listada é a utilização de estradas brasileiras realísticas para treinamento da arquitetura da rede neural.

Como já mencionado no Tópico 3.1, ter uma rede neural treinada com um banco de dados mais próximo da realidade em que o carro autônomo irá funcionar tende a permitir um melhor desempenho da inteligência artificial e, portanto, um melhor desempenho do carro autônomo.

3.4.2 ARQUITETURA DA REDE NEURAL

Como já levantado no Tópico 3.2, algumas arquiteturas de redes neurais já foram testadas com o intuito de desenvolver carros autônomos. Neste tópico são destacadas as diferenças e contribuições da arquitetura desenvolvida neste trabalho quando comparada às já previamente publicadas.

Em [25], duas arquiteturas baseadas em aprendizagem por reforço e uma baseada em aprendizado supervisionado são desenvolvidas e comparadas entre si. A arquitetura baseada em aprendizado supervisionado testada em [25] possui como entrada para a rede neural apenas uma imagem e fornece como saída somente o ângulo da direção. Diferentemente de [25], a rede neural desenvolvida nesta pesquisa recebe dados mistos de imagens e valores numéricos (como a velocidade atual do veículo) e é capaz de controlar não só o ângulo da direção, mas também a aceleração e velocidade do veículo, como é detalhado no capítulo de Metodologia (Capítulo 4). Sendo assim, a arquitetura desenvolvida nesta pesquisa trata o problema de automação veicular de forma mais completa do que a desenvolvida em [25]. Desta forma, uma segunda contribuição listada é o desenvolvimento de uma arquitetura que recebe dados mistos de imagens e valores numéricos.

Como o ambiente virtual utilizado em [25] não foi disponibilizado pelos autores, não é possível fazer a comparação da arquitetura desenvolvida neste trabalho com as arquiteturas por reforço desenvolvidas por [25], já que o único modo de fazer esta comparação é observando o desempenho do veículo no ambiente utilizado pelo autor.

Em [27] e [32], também é utilizada uma arquitetura com apenas uma imagem de entrada e o ângulo da direção como saída e nenhum controle de velocidade é feito. [26], apesar de utilizar *Self-Driving Car* como título, no trabalho somente é realizada uma classificação de imagens de uma maquete retornando três classes: em frente, curva à esquerda ou curva à direita. Apenas com essa saída não é possível conduzir um veículo de forma autônoma, já que o ângulo da direção, característica imprescindível para o controle do carro, não é previsto pela rede neural.

Um ambiente virtual é utilizado e uma arquitetura de *Deep Q Network* foi desenvolvida por [28]. A rede neural recebe uma imagem como entrada e retorna uma entre 3 ações: virar 10 graus para a esquerda, virar 10 graus para a direita ou seguir em frente. Tendo em vista que apenas uma das ações é executada por vez, isso torna a condução de veículo real impossibilitada, já que para efetuar uma curva, por exemplo, é necessário atuar na velocidade e no ângulo de direção simultaneamente.

Uma arquitetura baseada em CNN e LSTM foi criada por [33], mas apenas o ângulo de direção é retornado pela rede neural, e como já mencionado, se faz necessária a velocidade e o ângulo de direção para o funcionamento adequado de um veículo autônomo.

Por fim, em [34], um jogo de corrida foi utilizado como ambiente virtual e uma CNN foi criada para processar a imagem do jogo. Porém, a rede neural só é capaz de prever o ângulo da direção, se enquandrando no mesmo problema relatado nos trabalhos [33] e [28].

4 METODOLOGIA

Neste capítulo são descritos os materiais e métodos utilizados neste trabalho. Como já citado na Seção 3.4, as contribuições desta pesquisa envolvem dois grandes aspectos do contexto de inteligência artificial: criação do banco de dados e a arquitetura da rede neural. Inicialmente, para a criação do banco de dados, foram coletados dados do ambiente virtual (ETS2) via teclado, porém, conforme detalhado no Capítulo 5, o desempenho da rede neural não foi satisfatório devido a problemas relativos a este tipo de coleta, como por exemplo, oscilação da direção. Com o intuito de minimizar este tipo de problema, a coleta de dados foi novamente realizada utilizando o volante *Logitech G920 Racing Wheel*, além de alguns outros processamentos nas imagens coletadas que ajudaram no desempenho da rede neural proposta, descritos com mais detalhes no subtópico 4.1.2. Desta forma, esta seção é dividida da maneira que se segue: **coleta e construção do banco de dados**, dividida em primeira fase da coleta, utilizando o teclado, e segunda fase da coleta, realizada por meio do volante; e **modelagem da rede neural**, em que são descritas as arquiteturas testadas para estes dois tipos de coleta.

4.1 COLETA E CONSTRUÇÃO DO BANCO DE DADOS

A metodologia de coleta dos dados para construção do banco foi dividida em duas fases. Na primeira fase, foi utilizado o teclado como dispositivo de entrada para o jogo ETS2 e a tela padrão do jogo foi utilizada. Na segunda fase, foi utilizado um volante como dispositivo de entrada para o jogo ETS2 e a câmera frontal do jogo foi utilizada. Este modo de câmera permite que a detecção de faixa e outros processamentos que serão discutidos a seguir sejam executados de maneira mais fácil por causa do ângulo de visão.

Nas duas fases as seguintes tarefas foram executadas: coleta das imagens, coleta dos valores numéricos, construção do arquivo CSV (arquivo que contém o banco de dados para treinamento da rede neural) e pré-processamento dos dados. O detalhamento de cada fase será dado nos tópicos que seguem.

4.1.1 PRIMEIRA FASE DE COLETA DE DADOS

Nesta fase, todos os dados foram coletados a partir da condução do veículo no jogo ETS2 utilizando o teclado.

4.1.1.1 COLETA DAS IMAGENS

Na Figura 4.1, é apresentada a imagem padrão do jogo *Euro Truck Simulator 2*, que serviu como base para definir as regiões de interesse que serão utilizadas como entrada para a rede neural. Como pode ser visto na Figura 4.2, tais regiões de interesse são mostradas e é possível observar que estas incluem: visão frontal da pista, retrovisores e o GPS, destacadas por meio de um quadrado vermelho.

Figura 4.1 – Imagem padrão do jogo Euro Truck Simulator 2.

Figura 4.2 – Regiões de interesse (visão frontal da pista, retrovisores e o GPS) na imagem padrão destacados por meio de quadrados vermelhos.

Inicialmente, foi montada uma única imagem agrupando todas as regiões de interesse. O resultado desse agrupamento pode ser visto na Figura 4.3. O retrovisor da

esquerda foi rotacionado em -90 graus e o da direita em 90 graus. Essa diferença de sentido na rotação foi feita com o intuito de facilitar o aprendizado da rede e possibilitar a detecção de veículos em ambos os retrovisores de forma diferenciada. Já que, se esta rotação não fosse feita, os padrões (bordas e formatos) encontrados nos veículos em ambos os retrovisores seriam os mesmos.

Figura 4.3 – Imagem final montada a partir das regiões de interesse.

Utilizando uma arquitetura de rede neural preliminar, percebeu-se que a utilização da imagem mostrada na Figura 4.3 como entrada para a rede neural não demonstrou bons resultados (o carro saía da faixa em menos de 2 segundos de jogo, necessitando de intervenção humana), pois os filtros que deveriam aprender padrões de cada região de interesse tinham influência do restante da imagem, dificultando sua aprendizagem. A partir desta problemática, optou-se por salvar cada região de interesse em um arquivo separado e desenvolver uma arquitetura de redes neurais convolucionais com múltiplas imagens de entrada.

4.1.1.2 COLETA DOS VALORES NUMÉRICOS

As principais variáveis numéricas de interesse são: o ângulo da direção, a velocidade e a aceleração do veículo. O ângulo da direção e a aceleração foram coletadas a partir de um script capaz de emular o volante G27 da Logitech, o qual convertia comandos do teclado nas teclas: I (aceleração), K (frenagem), J (curva para esquerda) e L (curva para direita) em valores de ângulo de direção e aceleração. Os valores de ângulo variam entre -32767 até 32767, em que: 0 representa a direção estática, -32767 a direção rotacionada totalmente para a esquerda e 32767 a direção rotacionada totalmente para a direita. Os valores de aceleração variam de 0 a 65535, em que: 32768 representa o pedal estático (sem aceleração e sem frenagem), valores maiores que 32768 indicam aceleração positiva e valores menores que 32768 indicam aceleração negativa (frenagem).

A região da tela do jogo que mostra o velocímetro é apresentada na Figura 4.4. Esta região poderia ter sido extraída na etapa anterior para ser passada como uma das entradas da rede neural. Porém, no intuito de facilitar o aprendizado da rede, foi feito um script capaz de processar as imagens da velocidade, o qual recebe a imagem como entrada e retorna a velocidade em um valor inteiro de 0 a 90. O script extrator da velocidade

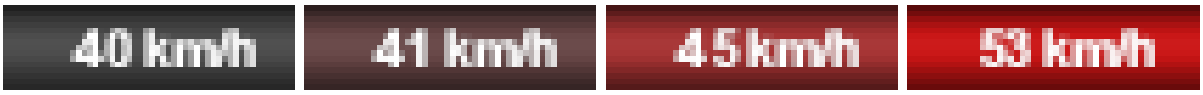
funciona da seguinte forma: uma imagem para cada valor de velocidade foi extraída do jogo, convertida para um array e salva em um arquivo. Então, uma vez que o script coleta a região contendo a velocidade, tal imagem é convertida em um array e comparada com todos os arrays salvos. O array que retornar a menor diferença é o array da velocidade atual do veículo.

Figura 4.4 – Velocímetro marcando 38 km/h.

A imagem que mostra a velocidade permanece com o fundo preto enquanto não ultrapassa o limite de velocidade da estrada na qual o veículo se encontra, como pode ser visualizado na Figura 4.4. Após ultrapassar o limite estabelecido se torna gradualmente mais avermelhada, como pode ser visto na Figura 4.5, na qual quatro valores de velocidade são mostrados para uma via com limite de velocidade de 40 km/h.

Figura 4.5 – Velocímetro em uma via com limite de velocidade de 40 km/h.

(a) (b) (c) (d)



Dessa forma, um pixel da região que muda a coloração é coletado e o valor relativo ao canal *red* desse pixel é extraído. Esse valor varia de 79 (quando não se está acima do limite) até 214 (quando a velocidade está muito acima do limite) e é fornecido como entrada para a rede neural, como detalhado no Tópico 4.2.

4.1.1.3 CONSTRUÇÃO DO ARQUIVO CSV

O script de coleta dos dados obteve um desempenho de 33 instâncias coletadas por segundo. Essas instâncias são armazenadas em um arquivo CSV de acordo com a estrutura apresentada na Tabela 4.3. É possível observar uma leve variação negativa na coluna Ângulo, que representa uma curva sutil para a esquerda.

A coluna ID recebe um *timestamp* com precisão de milisegundos. A coluna Excesso de Vel. recebe o valor do pixel do fundo do velocímetro, como mencionado anteriormente. As colunas I, K, J e L indicam quais teclas estão pressionadas no momento da coleta da instância atual, em que: 0 indica que a tecla não está pressionada e 1 indica que está. A partir dos valores das colunas apresentadas na Tabela 4.3, é possível observar que as teclas I e J foram pressionadas, caracterizando uma curva para a esquerda. Outra situação interessante a ser notada a partir dos dados da Tabela 4.3 é que no ID 1559699319233 tem-se o menor valor de ângulo da configuração apresentada e após isso, com o valor de J

em zero, o valor do ângulo começa a aumentar novamente até chegar em zero, caracterizado pelo retorno automático da direção para a posição de início.

Tabela 4.1 – Exemplo do arquivo CSV gerado na primeira fase de coleta de dados.

ID	Ângulo	Aceleração	Velocidade	Excesso de Vel.	I	K	J	L
1559699318983	0	60000	26	79	1	0	0	0
1559699319022	0	60000	26	79	1	0	0	0
1559699319060	0	60000	26	79	1	0	0	0
1559699319088	-85	60000	26	79	1	0	1	0
1559699319116	-340	60000	26	79	1	0	1	0
1559699319146	-595	60000	26	79	1	0	1	0
1559699319176	-765	60000	26	79	1	0	1	0
1559699319206	-1020	60000	26	79	1	0	1	0
1559699319233	-1190	60000	26	79	1	0	1	0
1559699319260	-890	60000	26	79	1	0	0	0
1559699319296	-590	60000	26	79	1	0	0	0
1559699319326	-290	60000	26	79	1	0	0	0
1559699319355	0	60000	26	79	1	0	0	0
1559699319384	0	60000	26	79	1	0	0	0

4.1.1.4 PRÉ-PROCESSAMENTO DOS DADOS

As colunas Ângulo e Aceleração foram normalizadas no intervalo de -1 até 1. As colunas Velocidade e Excesso de Vel. foram normalizadas no intervalo de 0 a 1. Duas novas colunas foram criadas: uma para indicar que o veículo estava sem aceleração, ou seja, nem I e nem K estavam sendo pressionados, e outra para indicar que o veículo estava em rota retilínea, ou seja, nem J e nem L estavam sendo pressionados. Essas duas colunas adicionais foram criadas com a ideia de construir um modelo de rede neural com duas saídas de classificação, uma para decidir entre acelerar, frear ou fazer nada e outra para decidir entre dobrar para a esquerda, dobrar para a direita ou se manter em linha reta.

Não houve a necessidade de utilizar técnicas de aumento de dados, já que 28 mil instâncias foram coletadas, totalizando cerca de 750 MB em imagens.

4.1.2 SEGUNDA FASE DE COLETA DE DADOS

A segunda fase de coleta foi feita com o intuito de gerar dados que possibilitem um treinamento mais efetivo da rede neural. A imagem do GPS foi otimizada para conter apenas a informação do caminho a ser seguido. A imagem da pista foi processada para passar dados tabulares para a rede neural em vez de passar a imagem bruta da pista, como por exemplo: distância para o veículo da frente, presença de sinal vermelho, centralização do veículo na faixa, entre outros. Além disso, todos os dados foram coletados a partir da condução do veículo no jogo ETS2 utilizando o volante com o intuito de remover a oscilação presente nas arquiteturas treinadas durante a primeira fase.

4.1.2.1 COLETA E PROCESSAMENTO DAS IMAGENS

Na Figura 4.6, é apresentada a imagem da câmera frontal do jogo *Euro Truck Simulator 2*, que serviu como base para definir as regiões de interesse que serão utilizadas como entrada para o treinamento da rede neural proposta nessa segunda fase. Como pode ser visto na Figura 4.4, tais regiões de interesse são mostradas e é possível observar que estas incluem: visão frontal da pista, retrovisores e o GPS, destacadas por meio de um quadrado vermelho.

Figura 4.6 – Câmera frontal do jogo Euro Truck Simulator 2.

4.1.2.1.1 Detecção dos limites de faixa

O primeiro passo para detectar os limites da faixa é cortar a imagem do jogo para obter a metade inferior, que é onde a estrada está localizada. O resultado deste procedimento pode ser visto na Figura 4.7 e esta imagem servirá como entrada para o próximo passo.

Figura 4.7 – Imagem cortada na pista.

O segundo passo é calibrar os limiares de valores usando as representações RGB (Figura 4.8), HSL (Figura 4.9) e YUV (Figura 4.10). O processo de calibração dos limiares

foi realizado manualmente de forma a destacar as faixas nas imagens de saída. Além disso, o filtro Sobel diagonal esquerdo e o filtro Sobel diagonal direito foram calculados e são mostrados nas figuras 4.11a e 4.11b. Os filtros S, U e V foram descartados porque não obtiveram informações relevantes relacionadas às faixas, como pode ser visto nas Figuras 4.9c, 4.10b e 4.10c, respectivamente. Os filtros restantes foram combinados para trabalharem juntos como um único filtro detector de faixa.

Figura 4.8 – Filtros RGB com limiares calibrados.

(a) Canal Red (b) Canal Green (c) Canal Blue

Figura 4.9 – Filtros HLS com limiares calibrados.

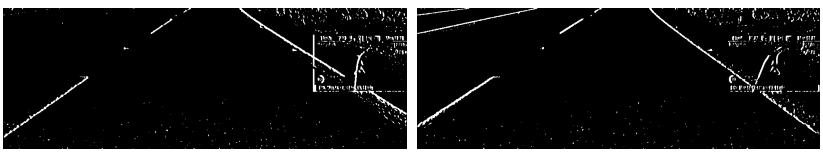
(a) Canal Hue (b) Canal Lightness (c) Canal Saturation

Figura 4.10 – Filtros YUV com limiares calibrados.

(a) Canal Y (b) Canal U (c) Canal V

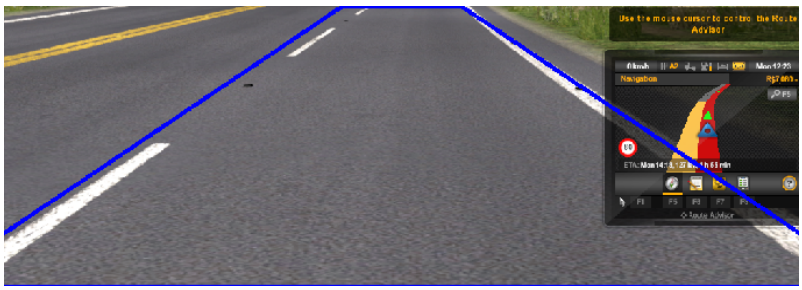
Figura 4.11 – Filtros Sobel com limiares calibrados.

(a) Diagonal esquerda (b) Diagonal direita



O terceiro passo é inverter a perspectiva da imagem de entrada para facilitar o processamento. Este procedimento é realizado selecionando manualmente quatro pontos na imagem de entrada, como mostra a Figura 4.12 na mesma forma que uma estrada reta. Em seguida, a perspectiva inversa é aplicada para transformar a forma em um retângulo, como mostra a Figura 4.13a. Agora é obtida uma imagem como uma vista superior e suas faixas são quase paralelas.

Figura 4.12 – Imagem antes da transformação de perspectiva.



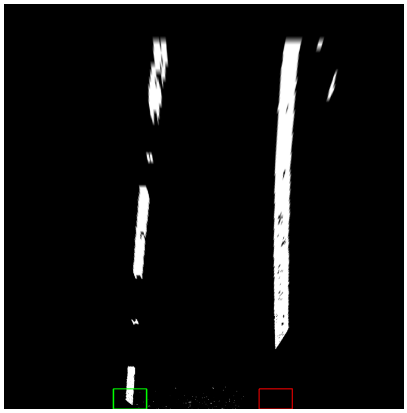
Os filtros selecionados na segunda etapa agora são aplicados na imagem distorcida e o resultado é mostrado na Figura 4.13b. Posteriormente, o meio histograma esquerdo e o meio histograma direito são calculados e seus valores máximos calculados. Esses valores serão usados como uma posição inicial horizontal para as janelas inferiores mostradas na Figura 4.13c.

Figura 4.13 – Processo de detecção de faixa.

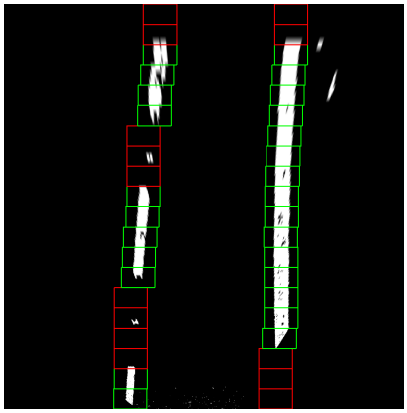
(a) Imagem distorcida.

(b) Imagem distorcida com filtros aplicados.

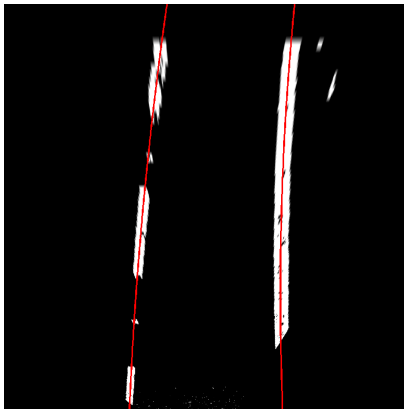
(c) Janelas da primeira linha.



(d) Todas as janelas detectadas.



(e) Resultado do Polyfit.



Se cada janela tiver uma quantidade mínima de pixels brancos (parâmetro definido pelo usuário), o centróide dos pixels brancos é calculado e a janela é movida de forma a centralizá-la no centróide calculado, também é exibido em verde indicando que a janela é

uma boa candidata a estar centrada na linha da pista. Se a janela não tiver a quantidade mínima de pixels brancos, ela é marcada como vermelha e será descartada posteriormente. Este procedimento é repetido para as próximas linhas, e a posição horizontal inicial é definida pela janela anterior. O resultado final é mostrado na Figura 4.13d.

Uma função polinomial de grau 2 é ajustada para os candidatos esquerdos e direitos. O resultado é mostrado na Figura 4.13e. A área entre as linhas é preenchida e a transformação da perspectiva é desfeita. A imagem agora é adicionada à primeira imagem de entrada e o resultado é mostrado na Figura 4.14. Para calcular a distância do veículo ao centro da pista, foi utilizada a Equação 4.1, que assume que a câmera frontal está centralizada.

$$vo = \left(\frac{w}{2} - \frac{(lx + rx)}{2}\right) * \frac{3,5}{500} \tag{4.1}$$

Na equação 4.1, vo significa deslocamento do veículo em metros, w significa a largura da imagem frontal, lx e rx são os valores de x dos polinômios ajustados na parte inferior da imagem (com y = altura da imagem frontal) e $3,5/500$ são constantes para converter pixels em metros em relação ao eixo x, aproximadamente. A distância estimada é salva no conjunto de dados CSV na coluna denominada *Centralização na Faixa* (ver Tabela 4.3).

Figura 4.14 – Resultado da detecção de faixa.

4.1.2.1.2 Detecção de veículo

A tarefa de detectar veículos e obstáculos na vista frontal pode ser facilitada usando apenas a área entre as faixas detectadas, como pode ser visto na Figura 4.15. Deste forma, um filtro binário pode ser usado para detectar as sombras. O resultado dessa filtragem pode ser visto na Figura 4.16. Se algum objeto é detectado, o objeto mais próximo é usado para limitar a zona de alcance livre de obstáculos, como mostra a Figura 4.17. A altura da área verde, que representa a zona de alcance livre, pode ser usada para estimar a distância do obstáculo mais próximo.

Figura 4.15 – Região de detecção de veículo.

Figura 4.16 – Limiar de detecção de veículo.

Figura 4.17 – Resultado da detecção de veículo.

Para facilitar a estimativa da distância, a Figura 4.16 é distorcida, para remover a perspectiva, e o número de pixels do fundo até o obstáculo mais próximo é convertido em metros usando a Equação 4.2, na qual a distância estimada representa o número de pixels da parte inferior ao objeto detectado, em que 20/500 é a constante estimada em metros por pixel em relação ao eixo y. A distância é salva no conjunto de dados CSV, na coluna denominada *Distância* (ver Tabela 4.3).

$$d = y * 20/500 \tag{4.2}$$

4.1.2.1.3 Detecção de semáforos

O primeiro passo para detectar semáforos no jogo ETS2 é obter a metade superior da imagem. O próximo passo é usar uma representação de cores HSV, na qual as cores podem ser separadas usando apenas um canal. As cores vermelho, amarelo e verde são definidas em diferentes faixas no canal de matiz (H). A cor vermelha varia do valor de

matiz 0 a 15 e 165 a 179. O próximo passo é detectar regiões com uma quantidade mínima de pixels na mesma cor. Se isso acontecer, a saída é 1 e, se não, a saída é 0. Essas saídas são salvas no arquivo CSV do conjunto de dados nas colunas denominadas Vermelho.

4.1.2.1.4 Otimização da imagem do GPS

Durante a primeira fase de coleta e modelagem foi notado que a rede neural não conseguia entender o percurso a ser seguido quando a imagem do GPS continha muitas avenidas. Por isso, a imagem do GPS precisou ser simplificada para ser passada para a rede neural. Esta simplificação pode ser vista na Figura 4.18, onde é mostrada a imagem original do GPS (Figura 4.18a) e a imagem processada (Figura 4.18b).

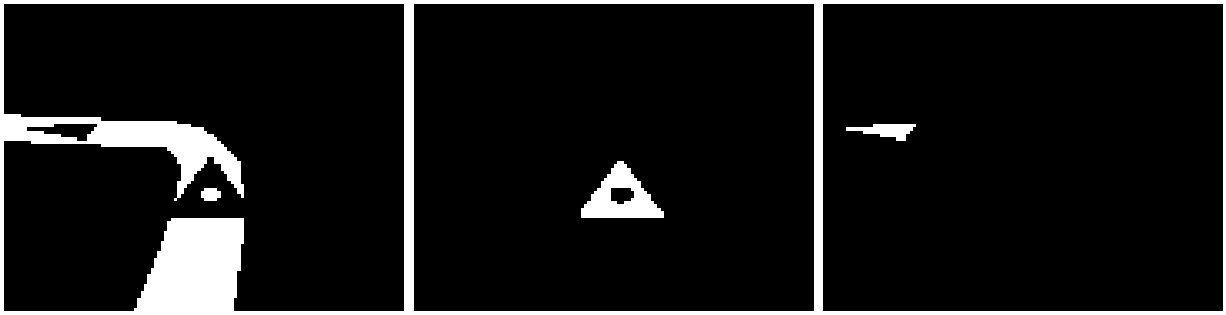
Figura 4.18 – Otimização da imagem do GPS.

(a) Imagem original. (b) Imagem otimizada.

Este procedimento foi feito utilizando três detectores de cores: vermelho, azul e verde, como mostrado na Figura 4.19. A soma dos três detectores é o resultado mostrado na Figura 4.18b. A imagem mostrada na Figura 4.18b é a única imagem utilizada pela rede neural desenvolvida durante a fase 2.

Figura 4.19 – Detecção de cores na imagem original do GPS.

(a) Detecção de Vermelho. (b) Detecção de Azul. (c) Detecção de Verde.



4.1.2.1.5 Classificação da qualidade da estrada

Como o foco do trabalho é a aplicação dos veículos autônomos em estradas brasileiras, foi preciso desenvolver um classificador da qualidade da pista para informar à

rede neural qual a condição da estrada atual. Com esta informação a rede neural pode ajustar a velocidade do veículo de acordo com a condição da estrada e assim evitar possíveis danos ao veículo.

Com a intenção de simplificar o classificador, optou-se por utilizar padrões binários locais, do inglês *Local Binary Pattern* (LBP), que é um algoritimo bastante utilizado para classificação de texturas [35].

Foram coletadas 5 imagens de estradas em boa qualidade (Figura 4.20) e 12 estradas em má qualidade (Figura 4.21) para o treinamento do algoritmo LBP.

Figura 4.20 – Exemplos de estradas com boa qualidade.

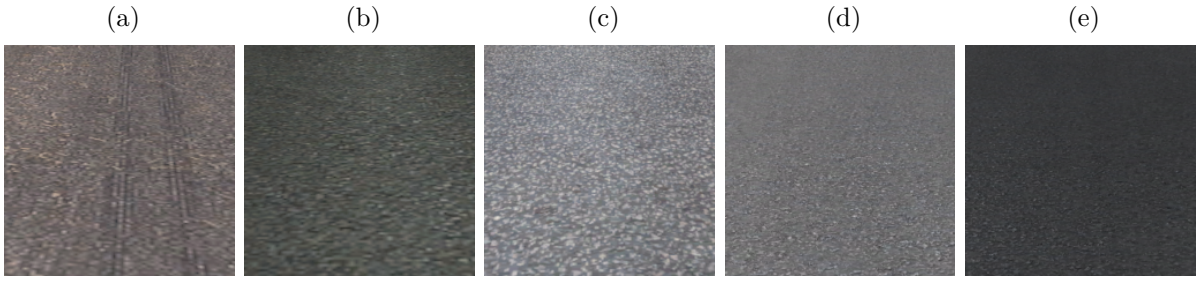
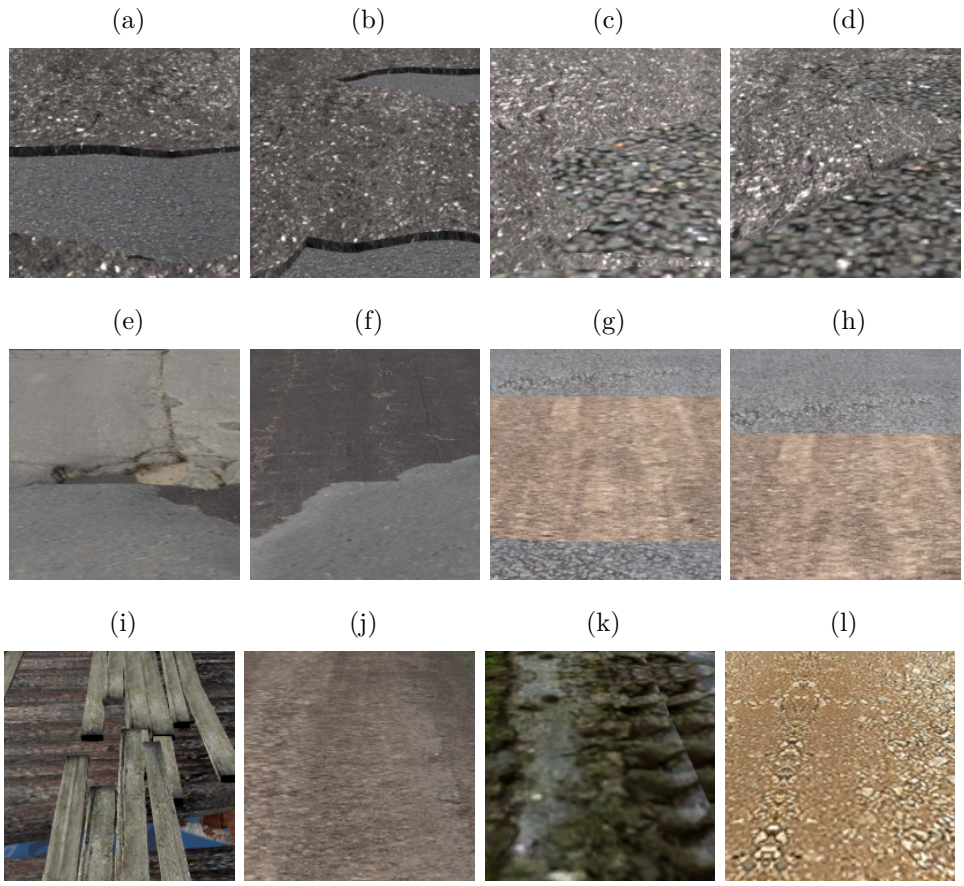


Figura 4.21 – Exemplos de estradas com má qualidade.



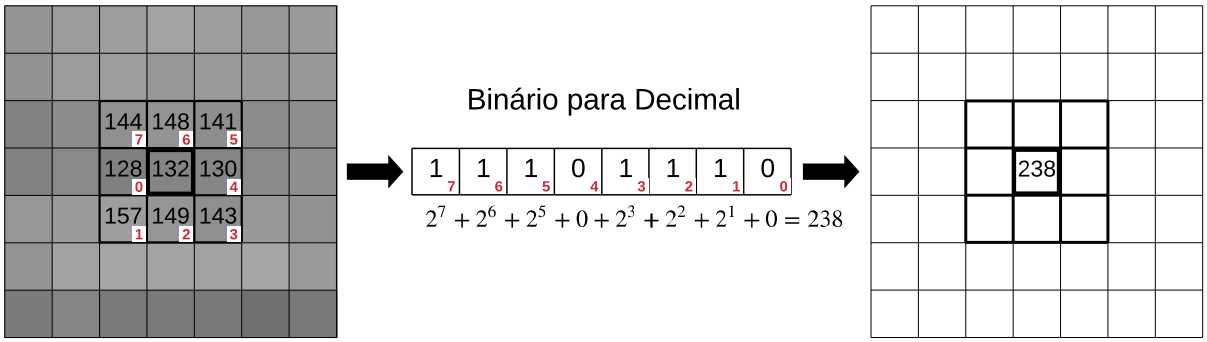
O treinamento do algoritmo LBP funciona com a execução das 4 etapas a seguir, para cada imagem da base de dados.

1. Carregamento da imagem original
   A partir da imagem original as etapas a seguir serão executadas.

2. Conversão para escala de cinza
   O algoritmo precisa da imagem em escala de cinza pois funciona para apenas um canal.

3. Cálculo da máscara LBP
   O cálculo da máscara LBP é feito pixel a pixel comparando os 8 pixels ao redor do pixel atual. Se o valor do pixel vizinho for maior ou igual ao pixel selecionado resulta no bit 1, caso contrário, bit 0. Com os 8 bits agrupados, tem-se o valor da máscara para o pixel selecionado. O procedimento do cálculo da máscara LBP é mostrado na Figura 4.22. É possível utilizar qualquer pixel vizinho como o bit mais significativo, porém, em todo o treinamento e teste, a ordem dos pixels vizinhos deve se manter a mesma.

Figura 4.22 – Cálculo da máscara LBP.



4. Cálculo do histograma LBP
   Uma vez que a máscara LBP foi calculada, seu histograma é calculado e normalizado, resultando em um array. Os arrays dos histogramas de todas as imagens devem ser armazenados, pois a classificação de uma nova imagem é feita comparando seu array com os arrays presentes na base de dados.

As Figuras 4.23 e 4.24 demonstram o processo do cálculo do histograma LBP para duas imagens, sendo a primeira de uma estrada em boa condição e a segunda de uma estrada em má condição. A qualidade da estrada durante a coleta de dados é salva no conjunto de dados CSV, na coluna denominada *Qual. da Estrada* (ver Tabela 4.3). Se a estrada for classificada como de boa qualidade, é salvo o valor 0. E se a estrada for classificada como de má qualidade, é salvo o valor 1.

Figura 4.23 – Cálculo do histograma LBP para estrada com boa qualidade.

(a) Original. (b) Escala de cinza. (c) Máscara LBP. (d) Histograma LBP.

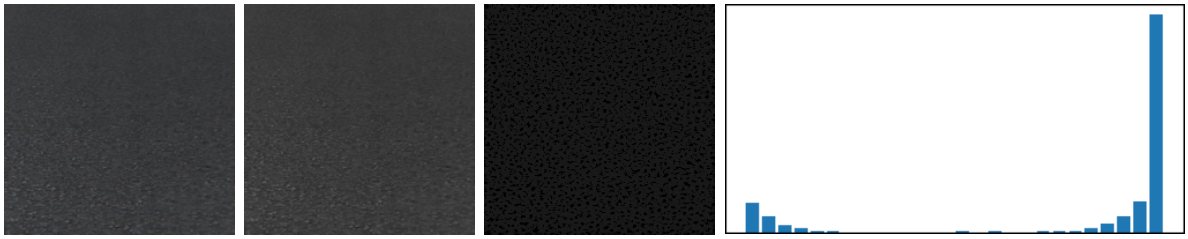
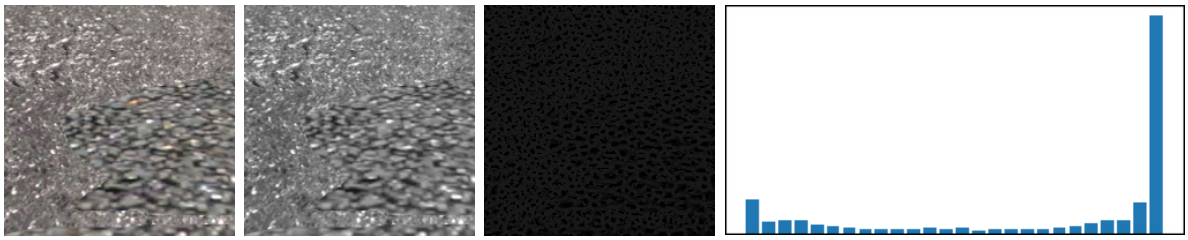


Figura 4.24 – Cálculo do histograma LBP para estrada com má qualidade.

(a) Original. (b) Escala de cinza. (c) Máscara LBP. (d) Histograma LBP.

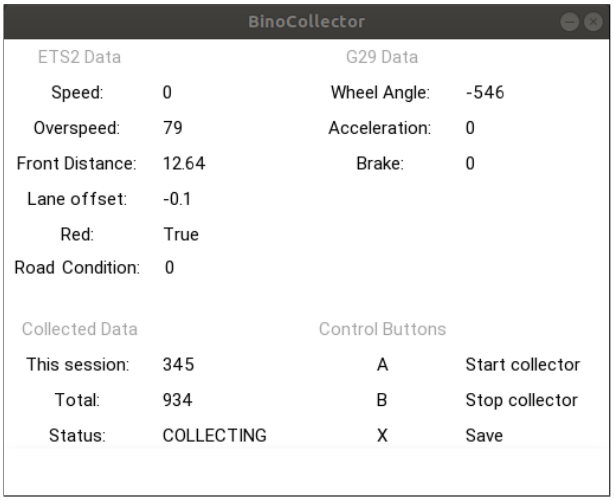


4.1.2.1.6 Software coletor de dados

Com o intuido de faciltar o trabalho na segunda fase de coleta, um coletor de dados, com adição de várias técnicas de processamento de imagens já descritas (ver Subtópicos 4.1.2.1.1 a 4.1.2.1.5), foi desenvolvido. O software é uma composição de duas janelas, uma com variáveis coletadas atualizadas em tempo real e outra com uma visualização da imagem do jogo mostrando a detecção da pista, semáforos e obstáculos, como pode ser visto na Figura 4.25.

Figura 4.25 – Resultado da Segunda Fase de Coleta.

(a) Interface do Script de Coleta. (b) Imagem com as Detecções aplicadas.



O software de coleta dos dados é controlado pelo botões A, B e X do volante, com as funções iniciar coletor, parar coletor e salvar dados, respectivamente. Quando o botão salvar dados é pressionado, a seção atual dos dados coletados é anexada ao conjunto de dados CSV e todas as imagens coletadas são movidas para o diretório de imagens.

4.1.2.2 AQUISIÇÃO DOS DADOS DO VOLANTE

A fim de coletar os dados do volante direto do dispositivo, é necessário entender o funcionamento do volante para realizar a aquisição dos dados. Como acontece em todos os dispositivos de entrada no Linux, um arquivo é criado no diretório /dev/input/ para ser gravado com todos os eventos do dispositivo. Para este dispositivo de entrada específico (volante Logitech G920), o arquivo criado para receber os eventos é /dev/input/js0, no qual os eventos são definidos em 8 bytes e cada botão e movimento no volante ou nos pedais dispara um evento diferente. Os bytes do evento foram decodificados e estão listados na Tabela 4.2. Alguns botões foram usados para controlar o aplicativo coletor, que será mostrado posteriormente, pois é mais rápido pressionar um botão no volante do que no teclado já que as mãos do condutor estão no volante.

Como o caminhão simulado está no modo automático, a embreagem não era necessária e nem os botões direcionais. Os eixos fornecem um valor de 2 bytes, que variam de -32767 a 32767 para o ângulo do volante e de 0 a 65535 para os pedais. Quando o volante está parado e centralizado, o valor de saída é 0, os valores negativos significam curva à esquerda e valores positivos significam curva à direita. Esses valores são salvos no conjunto de dados CSV nas colunas respectivamente denominadas Ângulo, Aceleração e Freio.

Tabela 4.2 – Eventos decodificados do volante G920.

Byte	**Relacionado a**	**Valor**	**Informação**
0	X	X	Não utilizado
1	X	X	Não utilizado
2	X	X	Não utilizado
3	X	X	Não utilizado
4	Botão	1	Botão Pressionado
	Botão	2	Botão Solto
	Eixo	0-255	Byte menos significativo
5	Eixo	0-255	Byte mais significativo
6	Botão	1	Relacionado ao Botão
	Eixo	2	Relacionado ao Eixo
7	Botão	0-10	Itendificação do Botão
	Eixo	0-5	Identificação do Eixo

O volante possui 6 eixos definidos pelo valor do byte de identificação do eixo:

- 0: Volante
- 1: Pedal de aceleração
- 2: Pedal de freio
- 3: Pedal de embreagem
- 4: Botões direcionais horizontais
- 5: Botões direcionais verticais

4.1.2.3 CONSTRUÇÃO DO ARQUIVO CSV

Um exemplo do conjunto de dados coletados na segunda fase é mostrado na Tabela 4.3. Esta tabela é formada pelos dados oriundos de cada etapa do processamento de imagens, descritos nos Tópicos 4.1.2.1.1 a 4.1.2.2. A coluna ID possui o registro de data e hora das instâncias, que podem ser usadas para recuperar os arquivos associados a ela, como a imagem do GPS. Os valores das colunas Ângulo, Acel. (Aceleração) e Freio são adquiridos a partir do volante (já descrito no Tópico 4.1.2.2). As colunas Vel. (Velocidade) e Excesso de Velocidade são obtidas a partir do processamento na imagem coletada já descrito no Tópico 4.1.1.2. Os valores das colunas Centr. (Centralização) na Faixa, Distância, Vermelho e Qual. (Qualidade) da Estrada são obtidos pelos processamentos nas imagens coletadas já descritos nos Tópicos 4.1.2.1.1, 4.1.2.1.2, 4.1.2.1.3 e 4.1.2.1.5, respectivamente.

Tabela 4.3 – Exemplo do Arquivo CSV Gerado na Segunda Fase de Coleta

ID	Ângulo	Acel.	Freio	Vel.	Excesso de Vel.	Centr. na Faixa	Distância	Vermelho	Qual. da Estrada
1559699318983	0	60000	0	26	79	0.31	347	0	0
1559699319022	0	60000	0	26	79	0.30	347	0	0
1559699319060	0	60000	0	26	79	0.30	347	0	0
1559699319088	-85	60000	0	26	79	0.27	347	0	0
1559699319116	-340	60000	0	26	79	0.27	347	0	0
1559699319146	-595	60000	0	26	79	0.26	347	0	0
1559699319176	-765	60000	0	26	79	0.26	347	0	0
1559699319206	-1020	60000	0	26	79	0.26	347	0	0
1559699319233	-1190	60000	0	26	79	0.29	347	0	0
1559699319260	-890	60000	0	26	79	0.29	347	0	0
1559699319296	-590	60000	0	26	79	0.26	347	0	0
1559699319326	-290	60000	0	26	79	0.24	347	0	0
1559699319355	0	60000	0	26	79	0.24	347	0	0
1559699319384	0	60000	0	26	79	0.23	347	0	0

Todas as colunas foram normalizadas para facilitar o treinamento da rede neural. Como as ações de acelerar e frear são mutuamente excludentes, essas variáveis precisaram ser unificadas. Esse procedimento foi necessário para evitar que a rede neural acione ambos os pedais simultaneamente. A coluna unificada possui o valor da aceleração subtraído

do valor do freio, dessa forma, valores negativos indicam frenagem e valores positivos aceleração.

4.2 MODELAGEM DA REDE NEURAL

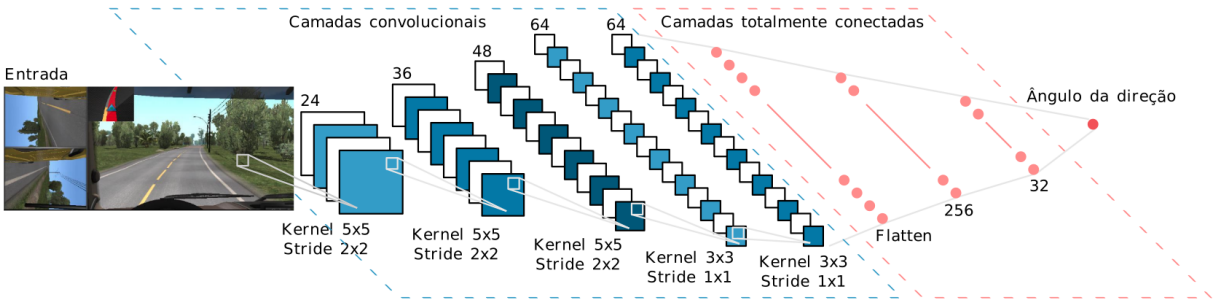
O processo de modelagem da rede neural se deu em duas fases. Na primeira fase foram utilizados os dados coletados utilizando o teclado. E na segunda fase foram utilizados os dados coletados utilizando o volante Logitech G920.

4.2.1 PRIMEIRA FASE DE MODELAGEM DA REDE NEURAL

A primeira fase de modelagem da rede neural se inicia com a implementação da arquitetura preliminar, criada para validar a imagem de entrada montada a partir do agrupamento das regiões (Figura 4.3), já mencionada no Tópico 4.1.1.1, que é apresentada na Figura 4.26. A ideia inicial foi tentar prever o valor do ângulo da direção com o uso de aproximadamente 8 mil instâncias coletadas. Esta arquitetura preliminar foi baseada na arquitetura desenvolvida pela Nvidia [36], que efetua a extração de características por meio de 5 camadas convolucionais, sendo as 3 primeiras com *stride* (passo do filtro) de dimensões 2 x 2 e *kernels* (tamanho do filtro) de dimensões 5 x 5 e as duas últimas com *stride* 1 x 1 e *kernels* 3 x 3. Após o aplainamento (*flattening*), a arquitetura original continha 3 camadas completamente conectadas com 100, 50 e 10 neurônios, respectivamente, e 1 neurônio na camada de saída. As 3 camadas completamente conectadas foram substituídas por duas camadas contendo 256 e 32 neurônios. Um detalhe interessante da arquitetura proposta pela Nvidia [36] é que não existem camadas de subamostragem, pois as camadas com *stride* 2 x 2 já desempenham este papel.

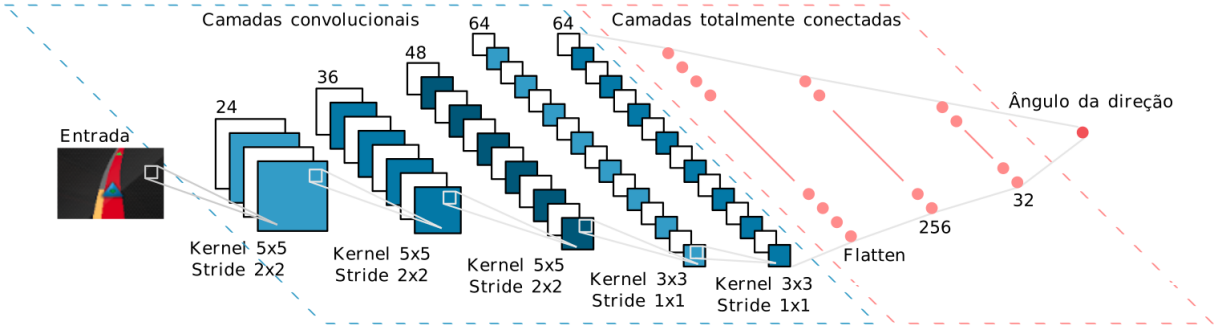
Dada a complexidade da imagem de entrada, pode-se levantar o seguinte questionamento: 8 mil instâncias são suficientes para o treinamento da rede? O questinamento surge tendo em vista que esta arquitetura não conseguiu realizar curvas, sendo uma das possíveis causas a quantidade de instâncias. Outra causa possível é a própria arquitetura.

Figura 4.26 – Arquitetura preliminar para validação da imagem de entrada.



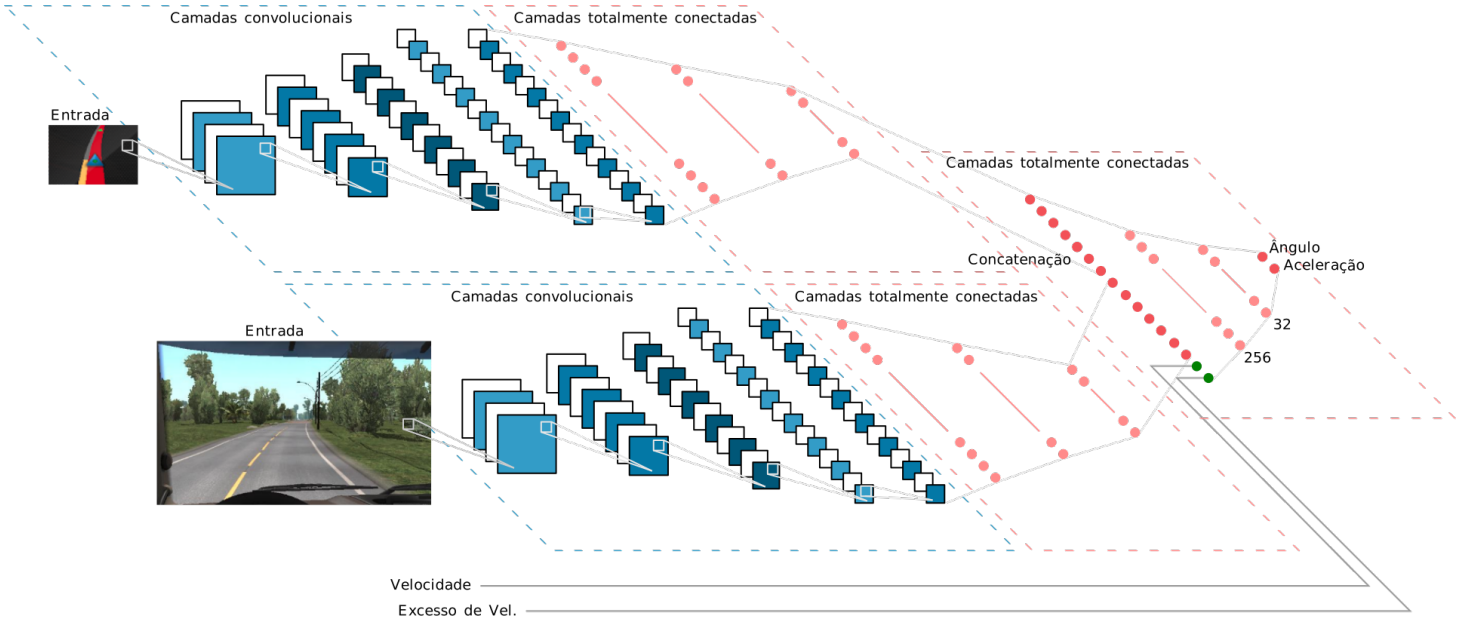
Devido ao desempenho não satisfatório da primeira arquitetura (Figura 4.26) e a fim de esclarecer se este resultado foi causado pela complexidade da imagem de entrada ou pela arquitetura em si, foi utilizado apenas a imagem do GPS como entrada (Figura 4.27). A nova imagem escolhida possui menor resolução e menos objetos. Dessa forma, é mais fácil controlar a seta do GPS no trajeto vermelho de forma isolada do que controlar por meio do processamento da imagem real da pista, minizando assim o problema. Cinco mil instâncias foram suficientes para a rede neural ser treinada e conseguir conduzir o veículo seguindo a rota do GPS, comprovando que a arquitetura funciona. No entanto, como nenhuma parte da visão frontal do veículo foi utilizada, a IA não executa ações perante outros veículos ou obstáculos.

Figura 4.27 – Arquitetura preliminar com utilização apenas da imagem do GPS.



No intuito de modelar uma arquitetura de redes neurais apta a processar múltiplas imagens e múltiplos dados numéricos simultaneamente, foi desenvolvida a arquitetura proposta na Figura 4.28.

Figura 4.28 – Arquitetura proposta na primeira fase de modelagem.



Nesta arquitetura, duas CNNs com a mesma configuração da arquitetura da Figura 4.27 são executadas em paralelo para processar a imagem do GPS e da visão

frontal separadamente. Em seguida, no lugar das CNN estarem gerando uma saída cada, suas penúltimas camadas (com 32 neurônios) são concatenadas gerando a camada de concatenação. Esta camada também recebe outras duas entradas numéricas via concatenação, que são a velocidade e o excesso de velocidade. A arquitetura ainda conta com mais 2 camadas totalmente conectadas com 256 e 32 neurônios e, por fim, a camada de saída com 2 neurônios, sendo um para prever o ângulo da direção e o segundo para prever o valor de aceleração.

A arquitetura mostrada na Figura 4.28 foi a que obteve os melhores resultados durante a primeira fase de modelagem. Outra arquitetura, baseada na camada de concatenação, mas com algumas modificações com relação a arquitetura anterior, foi desenvolvida e testada. Esta última arquitetura está exemplificada na Figura 4.29. A ideia por trás dela foi criar um agente inteligente capaz de conduzir o veículo utilizando o teclado, assim como o condutor humano na geração dos dados. Desse modo, a rede recebe o valor do ângulo da direção e da aceleração como entradas. Esses valores entram na camada de concatenação juntamente com a velocidade e excesso de velocidade. Outra diferença para a arquitetura da Figura 4.28, é que a saída não é mais uma regressão que prevê os valores do ângulo e aceleração (já que esses valores agora fazem parte da entrada da rede) e sim uma classificação dupla entre as teclas. A primeira classificação retorna uma opção entre as teclas J, L e Nenhuma, no intuito de retornar se o veículo vai decrementar o ângulo (J), incrementar o ângulo (L) ou manter o ângulo atual. A segunda classificação retorna uma opção entre as teclas I, K e Nenhuma, no intuito de retornar se o veículo vai incrementar a aceleração (I), decrementar a aceleração (K) ou manter a aceleração atual. Os incrementos e decrementos ocorrem em passos constantes de +250 e -250, respectivamente. Esta nova arquitetura testada obteve desempenho inferior quando comparada à arquitetura anterior.

4.2.2 SEGUNDA FASE DE MODELAGEM DA REDE NEURAL

A arquitetura proposta na segunda fase de modelagem da rede neural não utiliza a imagem da pista com as camadas convolucionais, como ocorreu nas arquiteturas da primeira fase. Nesta fase, são utilizados dados obtidos através do pré-processamento da imagem da pista como é possível observar na Figura 4.30. Ainda na Figura 4.30 pode-se observar que a imagem do GPS também é pré-processada antes de ser passada para a rede neural, como já descrito no Tópico 4.1.2.1.4.

4.3 CONSIDERAÇÕES FINAIS

Neste capítulo foram descritos os materiais e métodos utilizados neste trabalho. Os processos para coleta de dados nas duas fases de coleta foram descritos e apresentados.

Figura 4.29 – Arquitetura proposta na primeira fase com modificações.

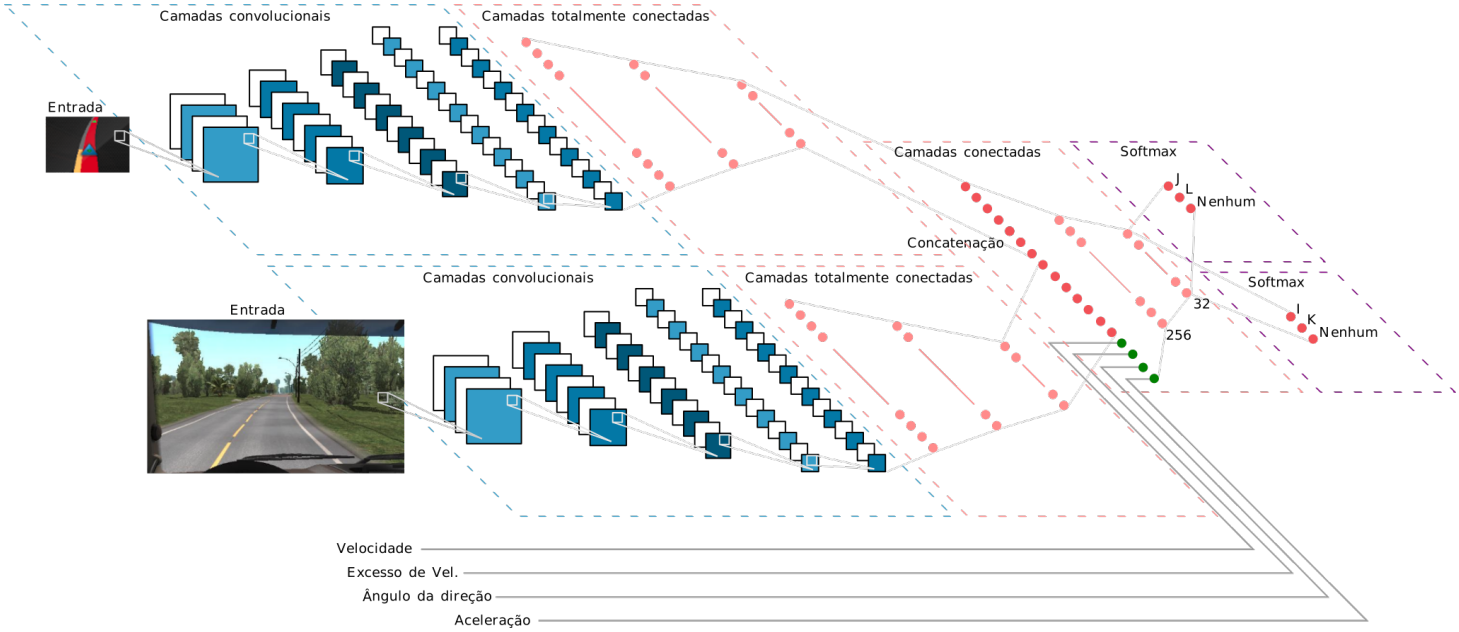
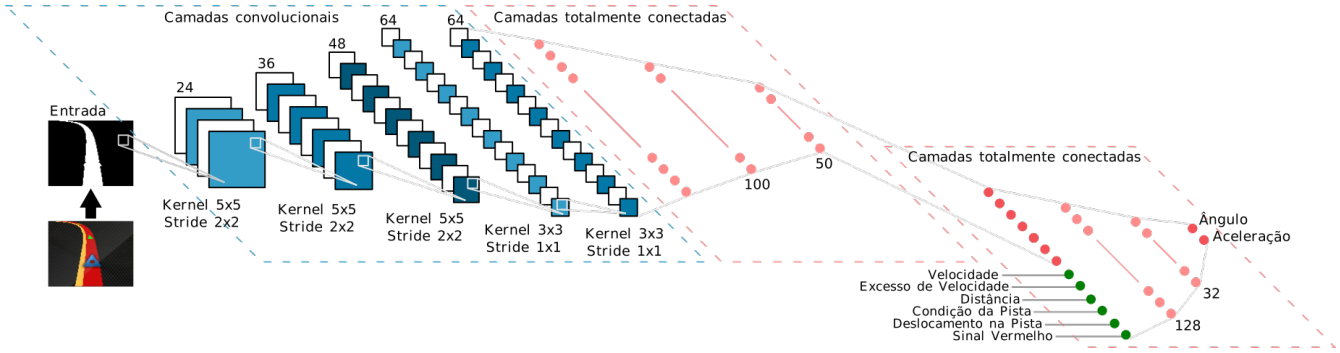


Figura 4.30 – Arquitetura proposta na segunda fase de modelagem.



De acordo com a pesquisa bibliográfica realizada neste trabalho, não foi encontrado nenhum trabalho sobre veículos autônomos que propõe uma arquitetura baseada na camada de concatenação. Esta camada permite uma maior flexibilidade para a rede neural, possibilitando a utilização de outros sensores, bastando apenas concatenar um neurônio para cada sensor adicional. Inclusive, outras imagens poderão ser utilizadas como entradas adicionais, as quais terão sua própria CNN, antes de entrarem na camada de concatenação, assim como foi realizado para as imagens do GPS e da visão frontal.

Todas as arquiteturas treinadas neste trabalho utilizaram a função de ativação ELU em todas as camadas convolucionais e também em todas as camadas totalmente conectadas. Uma camada de *dropout* com 50% de probabilidade de desconectar cada neurônio foi utilizada antes da camada de aplainamento em todas as redes convolucionais. Todas as arquiteturas também utilizaram a função de perda erro quadrático médio com o otimizador Adam e taxa de aprendizado 0,0001, com exceção da arquitetura baseada em *softmax* (Figura 4.29), que utilizou SGD (Descida de Gradiente Estocástico, do inglês,

Stochastic Gradient Descent) como otimizador e a função de erro Entropia Cruzada Binária (do inglês, *Binary Crossentropy*). Outros valores de taxa de aprendizado e outros tipos de função de ativação foram testados mas obtiveram resultados inferiores. Todas as redes foram treinadas com 100 épocas e *batch size* de 32 instâncias. Para evitar superajustamento (*overfitting*), o modelo treinado em cada época só era salvo se houvesse melhora em relação ao melhor modelo já salvo, além da utilização do *dropout* como citado anteriormente.

5 RESULTADOS

A inteligência artificial treinada na arquitetura proposta neste trabalho na primeira fase da modelagem, vista na Figura 4.28, foi capaz de conduzir o veículo do jogo *Euro Truck Simulator 2* na rota dada pelo GPS mesmo em estradas desconhecidas. Os dados coletados não continham estradas com falhas de infraestrutura, visto que, primeiro foi preciso conduzir o veículo de forma automática para depois partir para a identificação das falhas de infraestrutura e tomadas de decisão, trabalhados na segunda fase da modelagem.

Na Figura 5.1 é mostrado um trecho completamente desconhecido (sem participação no treinamento da rede neural), o qual apresenta quatro curvas. Todas as curvas foram executadas, porém, na curva 5.1c o veículo excedeu o limite direito da faixa como é possível observar pelo retrovisor direito. As curvas 5.1a e 5.1b foram realizadas em alta velocidade, 89 km/h e 86 km/h respectivamente, o que dificulta sua execução.

Como na primeira fase os dados foram gerados a partir da navegação humana utilizando um teclado e não um volante, alguns problemas foram encontrados. O primeiro problema é que não foi possível manter o ângulo da direção equivalente ao ângulo das curvas, pois as teclas incrementam ou decrementam o valor do ângulo em passos constantes. Então, em todas as curvas houve uma leve oscilação de ângulo por parte do condutor humano na geração dos dados. Este efeito se agrava com o aumento da velocidade e é possível observar que ele foi aprendido pela rede neural, pois em alguns trechos o volante apresenta uma oscilação visível. Então, pela dificuldade de condução do veículo a partir do teclado, é possível que na base de dados existam curvas nas quais o condutor humano excedeu o limite das faixas, o que explicaria o problema ocorrido na curva representada pela Figura 5.1c.

O segundo problema ocorrido na primeira fase é a variação dos valores de ângulo e aceleração, que são representados por um número inteiro de -32767 a 32767 para o ângulo e de 0 a 65535 para a aceleração. A variação desses valores não foi feita de forma contínua, já que as teclas atuam com incrementos ou decrementos em passos constantes. Isso dificulta o treinamento da rede, pois se trata de uma regressão na qual grande parte dos números do intervalo não constam nos dados. A base de dados possui apenas 2350 valores distintos de ângulo e 1453 valores distintos de aceleração de um intervalo de 65536 valores para ambas as variáveis.

Pelo histograma da variável ângulo, apresentado na Figura 5.2, pode-se observar que a maior parte das instâncias está com valor de ângulo zerado, resultado esperado pois

Figura 5.1 – Resultados da arquitetura trafegando em trecho desconhecido.

(a) Curva 1

(b) Curva 2

(c) Curva 3

(d) Curva 4

a maior parte do trajeto se dá em linha reta. E também pode ser visto que as variações de ângulo para a esquerda e para a direita tem o número de ocorrências similares.

Pelo histograma da variável aceleração, apresentado na Figura 5.3, pode ser observado que a maior parte das instâncias encontram-se com valor 60000, pois apesar do limite máximo ser 65536, o script gerador dos dados limitou a aceleração em 60000, para que a inteligência artificial não acelerasse de forma abrupta. A segunda maior parte encontra-se em 32768, que significa sem aceleração. Poucas instâncias possuem valores inferiores a 32768, ou seja, poucas frenagens foram feitas durante a coleta dos dados. Apesar disso, a rede neural consegue desacelerar antes das curvas ou quando ultrapassam o limite de velocidade. Esse conhecimento está dentro das instâncias com valores de aceleração entre 32768 e 60000.

De acordo com os resultados da primeira fase da modelagem, nas situações testadas foi necessário intervenção humana em momentos que havia curvas fechadas e obstáculos na via. Ainda assim a arquitetura desenvolvida tem potencial para controlar o carro em

Figura 5.2 – Histograma dos valores de ângulo encontrados na base de dados.

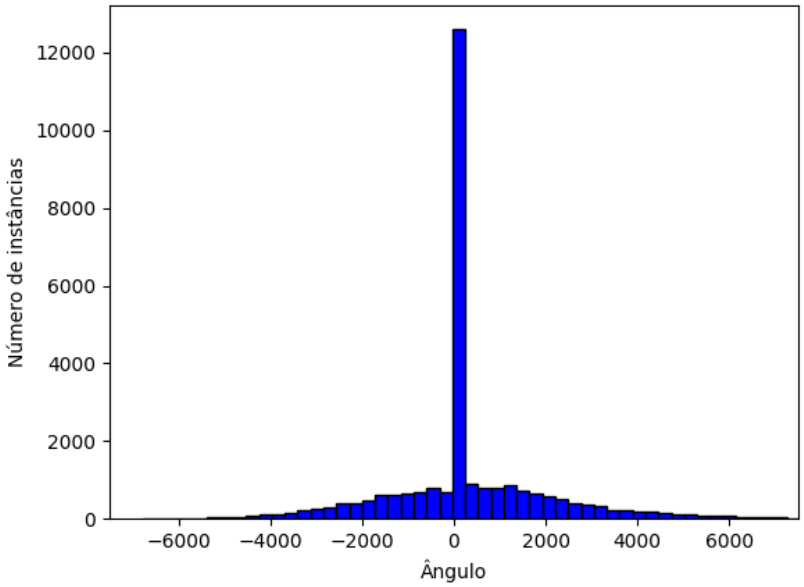
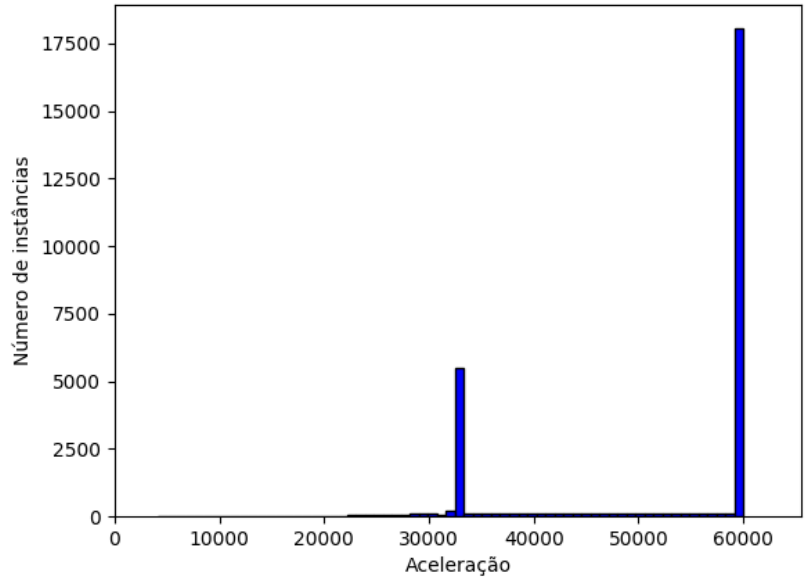


Figura 5.3 – Histograma dos valores de aceleração encontrados na base de dados durante a primeira fase.



estradas com infraestrutura degradada, desde que haja uma maior variação de situações na coleta de dados. Desta forma, na segunda fase de modelagem, dados com uma maior variedade de condições de estrada foram levados em consideração e os resultados são detalhados a seguir.

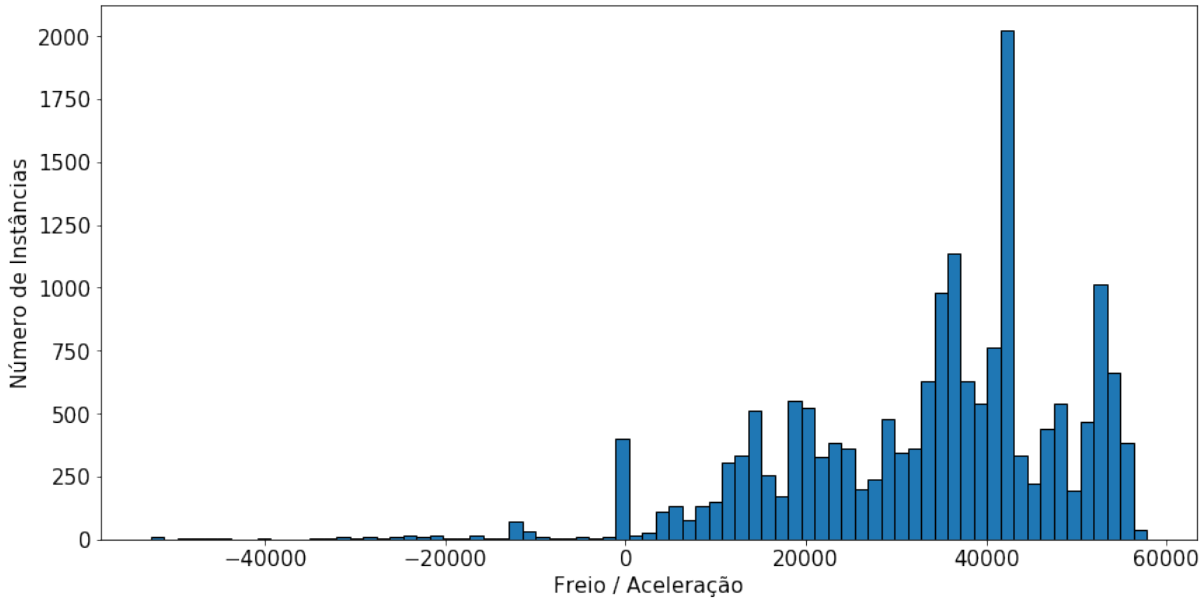
O classificador de qualidade da estrada funcionou de forma esperada possibilitando que a arquitetura desenvolvida na segunda fase fosse capaz de reduzir a velocidade do veículo quando uma estrada de má qualidade fosse detectada.

O problema de oscilação da direção foi minimizado devido ao uso do volante,

chegando a ficar imperceptível na maioria dos trechos. Com um número de instâncias coletadas de 17.676 (60% do número de instâncias coletadas na primeira fase) foram obtidos 4.098 valores distintos para a variável ângulo, quase o dobro do obtido na primeira fase, possibilitando uma regressão mais assertiva por parte da rede neural.

O uso do pedal para acelerar o veículo durante a segunda fase também possibilitou que a variável de aceleração ficasse mais representativa como pode ser visto na Figura 5.4. Isso possibilitou que a rede neural efetuasse um controle mais assertivo da velocidade.

Figura 5.4 – Histograma dos valores de aceleração encontrados na base de dados durante a segunda fase.



5.1 COMPARAÇÃO DE DESEMPENHO ENTRE A PRIMEIRA E A SEGUNDA FASE DE MODELAGEM

Com o intuito de comparar as duas arquiteturas propostas neste trabalho, 3 missões aleatórias do jogo ETS2 foram selcionadas e ambas as arquiteturas tentaram finalizar cada missão. Cada missão será detalhada nas subseções que seguem.

5.1.1 MISSÃO 01

A escolha da primeira missão está mostrada na Figura 5.5, sendo de Barro Branco até Tairaí. Esta é um missão de média duração e foi finalizada em 28 minutos por um jogador humano. Desses 28 minutos, 04 minutos são dentro da cidade e foram removidos das análises, tendo em vista que ambas as arquiteturas foram focadas na condução em estradas.

Figura 5.5 – Missão 01: Barro Branco para Tairaí.

Figura 5.6 – Trechos da missão 01.

(a) Trecho de barro sem sinalização.

(b) Curva acentuada.

(c) Veículo abandonado.

(d) Trecho com buracos.

Esta missão já começa com um trecho de barro e sem sinalização como pode ser visto na Figura 5.6a. Ambas as arquiteturas conseguiram passar desse trecho sem qualquer intervenção humana.

A missão possui uma curva acentuada que está mostrada na Figura 5.6b. A arquitetura da fase 1 não conseguiu realizar esta curva.

Uma parte do trecho da missão 01 possui um veículo abandonado na pista, como é possível observar na Figura 5.6c. A arquitetura da fase 1 não conseguiu identificar o veículo e bateu. Já a arquitetura da fase 2 conseguiu identificar o veículo e frear. Porém, ficou esperando que o veículo se movimentasse para continuar o trajeto.

O trecho final da missão 01 contém uma sequência de buracos como pode ser visto na Figura 5.6d. A arquitetura da fase 1 não conseguiu identificar os buracos e passou mutio rápido, com o veículo balançando. Já a arquitetura da fase 2 conseguiu identificar os buracos e reduzir a velocidade.

5.1.2 MISSÃO 02

A escolha da segunda missão está mostrada na Figura 5.7, sendo de Pariquera-Açu até Cajati. Esta é uma missão de curta duração e foi finalizada em 15 minutos por um jogador humano. Desses 15 minutos, 02 minutos são dentro da cidade e foram removidos das análises.

Figura 5.7 – Missão 02: Pariquera-Açu para Cajati.

Desconsiderando os trechos dentro da cidade, a missão inteira é com ruas asfaltadas e bem sinalizadas. Porém, possui um trecho com obras na pista, como mostrado na Figura 5.8 e ambas as arquiteturas bateram no bloqueio, provavelmente porque não havia estradas

com bloqueio na base de dados do treinamento de nenhuma fase. Outro problema que ocorreu nesta missão é que ela possui pedágio e as arquiteturas também não foram treinadas para parar no pedágio.

Figura 5.8 – Missão 02: Trecho em obras.

A partir da missão analisada, é possível concluir que a variabilidade de condições nas estradas é bem ampla. Dessa forma, uma nova melhoria no treinamento das arquiteturas pode ser realizada com a adição destas situações na fase de coleta de dados.

5.1.3 MISSÃO 03

A escolha da terceira missão está mostrada na Figura 5.9, sendo de Juquiá até Sorocaba. Esta é uma missão de média duração e foi finalizada em 26 minutos por um jogador humano. Desses 26 minutos, 03 minutos são dentro da cidade e foram removidos das análises.

A ponte presente na Figura 5.10a foi detectada apenas pela arquitetura da fase 2, na qual o veículo conduzido passou com a velocidade reduzida.

Os buracos mostrados na Figura 5.10b e as rachaduras mostradas na Figura 5.10c também foram detectados pela arquitetura da fase 2 e o veículo foi conduzido de forma mais suave do que o conduzido pela arquitetura da fase 1.

Figura 5.9 – Missão 03: Juquiá para Sorocaba.

Figura 5.10 – Trechos da missão 03.

(a) Trecho com ponte.

(b) Trecho com sequência de buracos.

(c) Trecho com rachaduras.

6 CONCLUSÃO

Neste trabalho foram desenvolvidas arquiteturas de redes neurais convolucionais para a condução de veículos autônomos em estradas degradas, com foco em estradas brasileiras, utilizando dados do jogo realístico *Euro Truck Simulator 2*. Para a aquisição dos dados de treinamento das arquiteturas desenvolvidas foram criados dois coletores de dados: um a partir do teclado e outro a partir do volante G290 Logitech. Desta forma, após a finalização desta pesquisa, pode-se elencar duas contribuições científicas: uma com relação à base de dados (estradas degradadas brasileiras) e os coletores, e a outra com relação à arquitetura da rede neural, a qual possui uma camada de concatenação que permite que diferentes tipos de dados sejam processados simultaneamente.

As arquiteturas desenvolvidas neste trabalho foram divididas em primeira fase da modelagem e segunda fase da modelagem, em que a primeira fase teve como dados de treino os dados coletados a partir do teclado, o que trouxe algumas questões a serem melhoradas na condução do veículo como, por exemplo, a oscilação da direção. A segunda fase de modelagem foi treinada com dados adquiridos a partir do volante G290 Logitech, conseguindo sanar alguns problemas como a oscilação da direção como também o fato de possuir os pedais permitiu uma melhor precisão para os dados adquiridos de frenagem e aceleração, conforme já discutido.

Os dados recebidos como entrada pela arquitetura da segunda fase de modelagem foram obtidos por meio da aplicação de várias técnicas de processamento de imagens, com o intuito de melhorar o desempenho da rede, como detecção de faixa, de veículos, de semáforos e a criação de um classificador da qualidade de estradas. Desta forma, a arquitetura da segunda fase de modelagem teve um desempenho melhor na condução de veículos no jogo *Euro Truck Simulator 2*, detectando estradas degradas e atuando de maneira assertiva gerando um tempo médio sem intervenção humana melhor do que a arquitetura da primeira fase de modelagem.

Apesar da arquiteuta da segunda fase de modelagem ter como desempenho um tempo médio sem intervenção humana de mais de 80% do tempo total da rota, algumas melhorias podem ser realizadas na coleta dos dados da segunda fase de modelagem como, por exemplo, balancear a base de dados; aplicar outras técnicas de processamento de imagens no classificador de estradas com o intuito de aumentar a variabilidade de estradas degradadas detectadas; aumentar o numero de condutores, filtrando erros de direção; bem como aumentar o número de instâncias adquiridas para treinamento da arquitetura.

REFERÊNCIAS

[1] B. L. H. W. e. Markus Maurer, J. Christian Gerdes, *Autonomous Driving: Technical, Legal and Social Aspects*, 1st ed. Springer-Verlag Berlin Heidelberg, 2016. [Online]. Available: http://gen.lib.rus.ec/book/index.php?md5=49799b3cf1411bdfe8e81ecfe032658b

[2] T. Litman, "Autonomous vehicle implementation predictions," https://www.vtpi.org/avip.pdf, Julho 2018.

[3] Q. Rao and J. Frtunikj, "Deep learning for self-driving cars: Chances and challenges," in *2018 IEEE/ACM 1st International Workshop on Software Engineering for AI in Autonomous Systems (SEFAIAS)*, May 2018, pp. 35–38.

[4] T. Okuyama, T. Gonsalves, and J. Upadhay, "Autonomous driving system based on deep q learnig," in *2018 International Conference on Intelligent Autonomous Systems (ICoIAS)*, March 2018, pp. 201–205.

[5] J. Henriksson, M. Borg, and C. Englund, "Automotive safety and machine learning: Initial results from a study on how to adapt the iso 26262 safety standard," in *2018 IEEE/ACM 1st International Workshop on Software Engineering for AI in Autonomous Systems (SEFAIAS)*, May 2018, pp. 47–49.

[6] D. Li, D. Zhao, Q. Zhang, and Y. Chen, "Reinforcement learning and deep learning based lateral control for autonomous driving [application notes]," *IEEE Computational Intelligence Magazine*, vol. 14, no. 2, pp. 83–98, May 2019.

[7] I. I. de Logística e Supply Chain, "Custos logísticos no brasil," http://www.ilos.com.br/ilos_2014/wp-content/uploads/PANORAMAS/PANORAMA_brochura_custos.pdf, 2014.

[8] R. Threlfall and R. ThoughtLab, "Autonomous vehicles readiness index: Assessing countries' openness and preparedness for autonomous vehicles," https://assets.kpmg/content/dam/kpmg/xx/pdf/2018/01/avri.pdf, 2018.

[9] M. Q. Lessa, "Anuário estatístico de transportes 2010 - 2016," http://www.transportes.gov.br/images/2017/Sum%C3%A1rio_Executivo_AET_-_2010_-_2016.pdf, 2017.

[10] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[11] M. A. Nielsen, "Neural networks and deep learning," http://neuralnetworksanddeeplearning.com, 2018.

[12] Églen da Veiga Protas, "Visualização de Camadas Intermediárias de Redes Neurais Convolucionais de Transformação de Imagem," Master's thesis, Universidade Federal do Rio Grande, 2017.

[13] Y. Kang, H. Yin, and C. Berger, "Test your self-driving algorithm: An overview of publicly available driving datasets and virtual testing environments," *IEEE Transactions on Intelligent Vehicles*, pp. 1–1, 2019.

[14] R. Guzmán, J.-B. Hayet, and R. Klette, "Towards ubiquitous autonomous driving: The ccsad dataset," in *Computer Analysis of Images and Patterns*, G. Azzopardi and N. Petkov, Eds. Cham: Springer International Publishing, 2015, pp. 582–593.

[15] G. Neuhold, T. Ollmann, S. R. Bulò, and P. Kontschieder, "The mapillary vistas dataset for semantic understanding of street scenes," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 5000–5009.

[16] Y. He, H. Yang, and S. Wang, "CDBV: A Driving Dataset With Chinese Characteristics From a Bike View," *IEEE Access*, vol. 7, pp. 51 714–51 723, 2019.

[17] R. Hussain and S. Zeadally, "Autonomous Cars: Research Results, Issues, and Future Challenges," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1275–1313, 2019.

[18] M. G. Bechtel, E. Mcellhiney, M. Kim, and H. Yun, "DeepPicar: A Low-Cost Deep Neural Network-Based Autonomous Car," in *2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, Aug. 2018, pp. 11–21.

[19] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to End Learning for Self-Driving Cars," *arXiv:1604.07316 [cs]*, Apr. 2016, arXiv: 1604.07316. [Online]. Available: http://arxiv.org/abs/1604.07316

[20] D. Dong, X. Li, and X. Sun, "A Vision-Based Method for Improving the Safety of Self-Driving," in *2018 12th International Conference on Reliability, Maintainability, and Safety (ICRMS)*, Oct. 2018, pp. 167–171.

[21] M. V. Smolyakov, A. I. Frolov, V. N. Volkov, and I. V. Stelmashchuk, "Self-Driving Car Steering Angle Prediction Based On Deep Neural Network An Example Of CarND Udacity Simulator," in *2018 IEEE 12th International Conference on Application of Information and Communication Technologies (AICT)*, Oct. 2018, pp. 1–5.

[22] J. Kim, G. Lim, Y. Kim, B. Kim, and C. Bae, "Deep learning algorithm using virtual environment data for self-driving car," in *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, Feb 2019, pp. 444–448.

[23] GitHub, "The state of the octoverse: machine learning," https://github.blog/2019-01-24-the-state-of-the-octoverse-machine-learning/, 2019.

[24] E. F. A. INÁCIO, "Mapa eldorado," https://mapaeldorado.com.br, 2019.

[25] S. Kardell and M. Kuosku, "Autonomous vehicle control via deep reinforcement learning," Master's thesis, Chalmers University of Technology, 2017.

[26] S. OwaisAli Chishti, S. Riaz, M. BilalZaib, and M. Nauman, "Self-driving cars using cnn and q-learning," in *2018 IEEE 21st International Multi-Topic Conference (INMIC)*, Nov 2018, pp. 1–7.

[27] X. Zhang, M. Chen, and X. Zhan, "Behavioral cloning for driverless cars using transfer learning," in *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, April 2018, pp. 1069–1073.

[28] T. Okuyama, T. Gonsalves, and J. Upadhay, "Autonomous driving system based on deep q learnig," in *2018 International Conference on Intelligent Autonomous Systems (ICoIAS)*, March 2018, pp. 201–205.

[29] J. Kim and C. Park, "End-to-end ego lane estimation based on sequential transfer learning for self-driving cars," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, July 2017, pp. 1194–1202.

[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[31] K. Gauen, R. Dailey, J. Laiman, Y. Zi, N. Asokan, Y. Lu, G. K. Thiruvathukal, M. Shyu, and S. Chen, "Comparison of visual datasets for machine learning," in *2017 IEEE International Conference on Information Reuse and Integration (IRI)*, Aug 2017, pp. 346–355.

[32] J. Kocić, N. Jovičić, and V. Drndarević, "Driver behavioral cloning using deep learning," in *2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH)*, March 2018, pp. 1–5.

[33] S. K. Saksena, N. B, S. Hegde, P. Raja, and R. M. Vishwanath, "Towards behavioural cloning for autonomous driving," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*, Feb 2019, pp. 560–567.

[34] S. Sharma, G. Tewolde, and J. Kwon, "Behavioral cloning for lateral motion control of autonomous vehicles using deep learning," in *2018 IEEE International Conference on Electro/Information Technology (EIT)*, May 2018, pp. 0228–0233.

[35] M. Quintana, J. Torres, and J. M. Menéndez, "A simplified computer vision system for road surface inspection and maintenance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 3, pp. 608–619, March 2016.

[36] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, and J. Zhang, "End to End Learning for Self-Driving Cars," *arXiv e-prints*, p. arXiv:1604.07316, Apr 2016.