Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Automóvel
Mestrado em Engenharia Automóvel

PROJETO AUTOMÓVEL

VERIFICAÇÃO FORMAL DE REDES NEURONAIS PROFUNDAS EM CONTEXTO DE CONDUÇÃO AUTÓNOMA

CARLOS ANDRÉ MACEDO GONÇALVES

Leiria, setembro 2024

Esta página foi propositadamente deixada em branco.

Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Automóvel
Mestrado em Engenharia Automóvel

PROJETO AUTOMÓVEL

VERIFICAÇÃO FORMAL DE REDES NEURONAIS PROFUNDAS EM CONTEXTO DE CONDUÇÃO AUTÓNOMA

CARLOS ANDRÉ MACEDO GONÇALVES
Número: 2223293

Projeto Automóvel realizado sob a orientação do Professor Doutor Luís Manuel Conde Bento (luis.conde@ipleiria.pt), Professor Doutor Diogo Nascimento Baptista (diogo.baptista@ipleiria.pt) e Professora Doutora Alexandra Nascimento Baptista (alexandra.nascimento@ipleiria.pt).

Leiria, setembro 2024

Esta página foi propositadamente deixada em branco.

AGRADECIMENTOS

Gostaria de começar por agradecer aos meus orientadores Professor Doutor Luís Manuel Conde Bento, Professor Doutor Diogo Nascimento Baptista e Professora Doutora Alexandra Nascimento Baptista por toda a orientação, apoio e prontidão, fatores fundamentais na elaboração deste projeto automóvel. Contribuíram partilhando o seu conhecimento, esclarecendo dúvidas e fornecendo os recursos necessários para o desenvolvimento deste projeto, contribuindo para o sucesso da mesmo.

Por fim, gostaria de agradecer à minha família por todo o suporte, apoio e incentivo durante este percurso.

Esta página foi propositadamente deixada em branco.

RESUMO

Ao longo deste projeto automóvel, é abordada a verificação formal de redes neuronais profundas (DNN) aplicadas à condução autónoma, com o objetivo de garantir a robustez e segurança desses modelos em situações críticas. A verificação formal é utilizada para avaliar as redes neuronais, visando prevenir falhas que possam resultar em riscos para a segurança, principalmente no contexto de veículos autónomos. Esta abordagem torna-se essencial, considerando o elevado impacto das decisões que as redes neuronais tomam ao classificar diferentes classes de sinais de trânsito, algo crucial para uma navegação segura.

Foram desenvolvidos, treinados e testados 3 modelos diferentes de redes neuronais convolucionais (CNN) capazes de classificar mais de 40 categorias diferentes de sinais de trânsito, utilizando a base de dados GTSRB, que contém cerca de 51839 amostras de imagens. A implementação foi realizada visando uma estrutura computacionalmente eficiente e robusta para potencial uso futuro em veículos autónomos. O trabalho também aborda a vulnerabilidade das redes neuronais a ataques adversariais, como o FGSM e o PGD, que podem levar a falhas na classificação de imagens de sinais de trânsito e, consequentemente, comprometer a robustez dos modelos. Para isso, foram aplicados métodos de verificação formal, que fornecem garantias matemáticas sobre a robustez e exatidão dos modelos desenvolvidos, mesmo quando expostos a um determinado conjunto de valores de perturbação.

Os resultados obtidos demonstram que, não obstante das dificuldades inerentes ao campo da verificação formal de DNN, é possível testar a robustez dos sistemas de classificação de sinais de trânsito, contribuindo para uma maior segurança no domínio da condução autónoma.

Este estudo contribui para o avanço na verificação formal de redes neuronais treinadas para a classificação de sinais de trânsito, destacando a importância de continuar a investigar soluções que garantam a segurança e a fiabilidade em sistemas críticos, como a condução autónoma.

Palavras-chave: Verificação Formal; Rede Neuronal Convolucional; Condução Autónoma; Propriedade de Robustez; Ataques Adversariais; MATLAB.

Esta página foi propositadamente deixada em branco.

ABSTRACT

Throughout this automotive project, the formal verification of deep neural networks (DNN) applied to autonomous driving is addressed, with the aim of guaranteeing the robustness and safety of these models in critical situations. Formal verification is used to evaluate neural networks in order to prevent failures that could result in safety risks, especially in the context of autonomous vehicles. This approach is essential given the high impact of the decisions that neural networks make when classifying different classes of traffic signs, which is crucial when it comes to a safe navigation.

Three different convolutional neural network (CNN) models capable of classifying more than 40 different categories of road signs were developed, trained and tested using the GTSRB database, which contains around 51839 image samples. The implementation was carried out with the aim of creating a computationally efficient and robust structure for potential future use in autonomous vehicles. The work also addresses the vulnerability of neural networks to adversarial attacks, such as FGSM and PGD, which can lead to failures in the classification of traffic sign images and, consequently, compromise the robustness of the models. To this end, formal verification methods were applied, which provide mathematical guarantees about the robustness and accuracy of the models developed, even when exposed to a given set of disturbance values.

The results obtained show that, despite the difficulties inherent in the field of formal verification of DNN, it is possible to test the robustness of traffic sign classification systems, contributing to a greater safety in the field of autonomous driving.

This study contributes towards advances in the formal verification of neural networks trained for traffic sign classification, highlighting the importance of continuing to investigate solutions that guarantee safety and reliability in critical systems, such as autonomous driving.

Keywords: Formal Verification; Convolutional Neural Network; Autonomous Driving; Robustness Property; Adversarial Attacks; MATLAB.

Esta página foi propositadamente deixada em branco.

ÍNDICE

LISTA DE FIGURAS

Esta página foi propositadamente deixada em branco.

LISTA DE TABELAS

Esta página foi propositadamente deixada em branco.

LISTA DE ACRÓNIMOS

ABS	*Anti-lock Braking System.*
ACC	*Adaptive Cruise Control.*
Adam	*Adaptive Moment Estimation.*
AdvRD	*Adversarial Raindrops.*
AI	*Artificial Intelligence.*
ASR	*Anti Slip Regulation.*
BAS	*Brake Assist System.*
BelgiumTS	*Belgium Traffic Signs.*
BIM	*Basic Iterative Method.*
BNN	*Binary Neural Network.*
CC	*Cruise Control.*
CNN	*Convolutional Neural Network.*
CPS	*Cyber-Physical Systems.*
DL	*Deep Learning.*
DNN	*Deep Neural Network.*
ELU	*Exponential Linear Unit.*
EOT	*Expectation Over Transformation.*
ESC	*Eletronic Stability Control.*
FFNN	*Feed-Forward Neural Network.*
FGSM	*Fast Gradient Sign Method.*
GAN	*Generative Adversarial Network.*
GPS	*Global Positioning System.*
GTSRB	*German Traffic Sign Recognition Benchmark.*
I-FGSM	*Iterative Fast Gradient Sign Method.*
LiDAR	*Light Detection And Ranging.*
LP	*Linear Programming.*
MCDNN	*Multi-Column Deep Neural Network.*
MILP	*Mixed Integer Linear Programming.*

ML	*Machine Learning.*
MLP	*Multilayer Perceptron.*
NNCS	*Neural Network Control Systems.*
NNV	*Neural Network Verification.*
ONNX	*Open Neural Network Exchange.*
OOD	*Out-of-Distribution.*
PGD	*Projected Gradient Descent.*
QCQP	*Quadratically Constrained Quadratic Program.*
RADAR	*RAdio Detection And Ranging.*
R-CNN	*Region-based Convolutional Neural Network.*
ReLU	*Rectified Linear Unit.*
ResNet	*Residual neural Network.*
R-FCN	*Region-based Fully Convolutional Network.*
RGB	*Red Green Blue.*
RGCE	*Regularized Guided Complement Entropy.*
RNN	*Recurrent Neural Network.*
RP2	*Robust Physical Perturbations.*
SAE	*Society of Automotive Engineers.*
SAT	*Boolean Satisfiability.*
SDP	*Semidefinite Programming.*
SELU	*Scaled Exponential Linear Unit.*
SGD	*Stochastic Gradient Descent.*
SMT	*Satisfiability Modulo Theories.*
SSD	*Single Shot Detector.*
TAA	*Targeted Attention Attack.*
Tanh	*Hyperbolic Tangent.*
TSC	*Traffic Sign Classification.*
TSD	*Traffic Sign Detection.*
TSR	*Traffic Sign Recognition.*
TT100K	*TsinghuaTencent 100K.*
VNN-COMP	*Verification of Neural Networks Competition.*
VNN-LIB	*Verified Neural Network Library.*

1

INTRODUÇÃO

Neste capítulo são indicadas as motivações que tornam relevante este estudo e descrito o enquadramento da verificação formal de redes neuronais profundas no contexto da condução autónoma. São ainda descritos os objetivos que se pretendem alcançar, bem como o âmbito do trabalho. Por fim, é apresentada a estrutura do projeto, de modo a uma melhor orientação durante a sua leitura e análise.

1.1 ENQUADRAMENTO E MOTIVAÇÃO

Nas últimas décadas, o estudo da tecnologia da Inteligência Artificial (que provém do inglês, *Artificial Intelligence* (AI)) e do *Machine Learning* (ML) está a tornar-se cada vez mais popular e relevante, uma vez que tem sido utilizada em muitos domínios diferentes com um elevado grau de sucesso. Entre estes, aqueles que são críticos para a segurança e a proteção, tal como a condução autónoma, o controlo de drones e robôs, os cuidados de saúde inteligentes, a segurança pública, a agricultura e aplicações de emergência, são frequentemente os que possuem um interesse peculiar para as comunidades de investigação e a indústria tecnológica [1].

No que concerne esta tese, o domínio crítico para a segurança e proteção em estudo é a condução autónoma dado que o surgimento e o aprofundamento, nos últimos anos, do conhecimento nesta área promete modernizar o setor automóvel e revolucionar a mobilidade urbana. Grandes fabricantes de automóveis, tais como a Tesla, a GM, a Ford, a BMW, a Mercedes-Benz e a Waymo/Google, já se encontram neste momento a construir e a testar ativamente este tipo de automóveis no mercado [2], sendo que existem diferentes aplicações da tecnologia da Inteligência Artificial (AI) e do *Machine Learning* (ML) na engenharia automóvel. Algumas dessas aplicações incluem a implementação de estratégias de manutenção preditiva (por exemplo, os dados operacionais que os veículos atualmente geram, são utilizados em técnicas de *Machine Learning* (ML) que visam identificar potenciais anomalias que possam resultar em falhas ou avarias), a prevenção de riscos (por exemplo, muitos fabricantes automóvel utilizam a tecnologia de monitorização ocular para detetar sinais de fadiga por parte dos condutores e evitar assim que adormeçam enquanto conduzem), uma experiência de condução personalizada, a assistência *on-road* (por exemplo, muitos veículos possuem o processamento de linguagem natural que converte o discurso

humano em linguagem digital, o que permite aos assistentes de voz inteligentes integrados nos veículos compreenderem e responderem aos comandos humanos), a previsão de tráfego e otimização de trajetória, e a perceção do meio ambiente do veículo, que engloba a deteção e classificação dos objetos que se encontram nas imediações do mesmo, tal como outros automóveis, pedestres, a faixa de rodagem, sinais de trânsito, entre outros [3]. Desta forma, os veículos autónomos dependem fortemente de algoritmos avançados de ML, particularmente de redes neuronais profundas (que provém do inglês, *Deep Neural Network* (DNN)), a fim de reconhecer o meio ambiente que os rodeia, para assim tomarem decisões e circularem de forma segura em ambientes complexos. Estas DNN são responsáveis pelo processamento de grandes quantidades de dados dos sensores integrados nos veículos autónomos, incluindo imagens de câmaras, sinais *Light Detection And Ranging* (LiDAR) e sinais de *RAdio Detection And Ranging* (RADAR), de modo a identificar e classificar objetos, prever os seus movimentos (para objetos não estáticos) e garantir uma condução segura.

Uma das aplicações essenciais das redes neuronais profundas na condução autónoma é nas tarefas de perceção, mais especificamente na classificação de sinais de trânsito, que é fundamental para uma navegação racional e em cumprimento com as regras de trânsito nas estradas. A classificação de sinais de trânsito envolve o reconhecimento e a categorização de vários sinais a partir de informações ou dados de imagens captadas por câmaras instaladas nos veículos autónomos. Dado o eventual risco de vida ou morte numa tomada de decisão nos veículos autónomos, é fundamental garantir a robustez e exatidão destas redes neuronais profundas. No entanto, apesar das capacidades impressionantes das DNN no domínio da condução autónoma, estas também suscitam preocupações significativas quanto à sua fiabilidade e robustez para mínimas alterações invisíveis ao olho humano. No contexto da condução autónoma, uma classificação incorreta de um sinal de trânsito pode levar a resultados catastróficos, incluindo acidentes de trânsito e perda de vidas [2]. Assim, é imperativo garantir a completude e a segurança dos modelos de redes neuronais profundas utilizados no reconhecimento e classificação de sinais de trânsito. Para tal, são usados métodos de verificação formal, que oferecem uma abordagem rigorosa que garante a robustez dessas redes, provando matematicamente as propriedades em estudo.

A principal contribuição deste trabalho é um repositório de código-fonte no GitHub que implementa o sistema descrito: `https://github.com/ipleiria-robotics/RTFVNN4TSCA`.

1.2 ÂMBITO E OBJETIVOS

Como a área da verificação formal de redes neuronais profundas ainda é relativamente recente no domínio da AI e do ML, ainda não existe um extenso número de algoritmos e ferramentas que permitam verificar a propriedade de robustez, mas é possível afirmar que já existe um número razoável de métodos desenvolvidos para este fim, que impulsionam a investigação deste tema. Um dos problemas da verificação formal de redes neuronais profundas é que, até à data, a maior parte dos métodos desenvolvidos são suportados em estruturas de linguagem *Python*, *C++* e *JavaScript*, tais como os softwares *PyTorch* e *TensorFlow*, sendo que em estruturas de linguagem matemática, tal como o software MATLAB, só existe um pequeno número de métodos. Em vista de continuar com o desenvolvimento deste tema em programas de linguagem matemática, e devido à própria familiaridade com esse tipo de software, então o software utilizado ao longo da realização do trabalho é o MATLAB [4].

O objetivo deste projeto é investigar e desenvolver, usando o software MATLAB, modelos de rede neuronal para classificação de sinais de trânsito capazes de classificar no mínimo 40 tipos diferentes de sinais de trânsito, que sejam robustos a ataques adversariais através da aplicação de métodos de verificação formal. Para treinar e testar estes modelos, é utilizada uma base de dados alemã de sinais de trânsito, que possui mais de 50000 imagens de sinais de trânsito compreendidas entre 43 classes ou categorias com frequências de classe desequilibradas, com o nome *German Traffic Sign Recognition Benchmark* (GTSRB) [5]. Os modelos desenvolvidos devem ser de baixa complexidade computacional e, simultaneamente, de bom desempenho, de forma a poder ser feita a inferência da rede neuronal em dispositivos embarcados de baixa capacidade computacional.

Desta forma, os objetivos definidos são:

- Identificar e estudar os métodos de verificação formal utilizados por diversos autores nos seus trabalhos referentes à robustez adversarial de redes neuronais profundas;
- Identificar e personalizar diferentes tipos de redes neuronais computacionalmente simples utilizadas no campo da classificação de imagens;
- Diferenciar e estudar os diferentes tipos de ataques adversariais e defesas aplicados a sistemas de classificação;
- Construir e desenvolver um sistema de rede neuronal (classificador) com a finalidade de classificar imagens de sinais de trânsito;

- Simular e avaliar a eficácia dos ataques adversariais, defesas e métodos de verificação formal analisados no modelo de rede neuronal elaborado para classificação de imagens de sinais de trânsito.

1.3 ESTRUTURA DO PROJETO

O trabalho está organizado da seguinte forma:

No Capítulo 2 são introduzidos os principais conceitos e técnicas que servem de suporte à realização deste trabalho.

O capítulo 3 contém a revisão da literatura efetuada, onde são identificadas e aprofundadas algumas abordagens de verificação formal e de ataques adversariais utilizadas recentemente na classificação de imagens de sinais de trânsito, assim como os diferentes modelos de rede neuronal aplicados no domínio da condução autónoma.

O capítulo 4 apresenta e explica o problema desafio deste projeto, tal como a base de dados de imagens e o modelo base utilizado.

No capítulo 5 é apresentado o trabalho realizado, isto é, são demonstradas as abordagens utilizadas no software MATLAB e os resultados experimentais obtidos.

Por fim, o capítulo 6 evidencia as principais conclusões do trabalho apresentado neste projeto, sendo que, adicionalmente, também são propostas algumas tarefas que poderão ser realizadas no futuro, como seguimento do trabalho realizado.

2

TECNOLOGIAS DE SUPORTE

Serve o presente capítulo para introduzir as principais técnicas e conceitos que servem de suporte ao desenvolvimento, avaliação e implementação de uma verificação formal de redes neuronais profundas (DNN) em contexto de condução autónoma.

2.1 REDES NEURONAIS

Neste capítulo, as redes neuronais irão ser definidas genericamente como grafos de operações sobre números reais. Na prática, a forma desses grafos, designada por arquitetura, não é arbitrária já que os investigadores e engenheiros constroem cuidadosamente a cada dia novas arquiteturas de modo a se adaptarem a novas tarefas que emergem no quotidiano.

Consequentemente, devido à sua grande variedade de utilidade e prestabilidade, as redes neuronais começam a ser cada vez mais utilizadas e a ter cada vez mais impacto em muitos setores da indústria. Alguns exemplos de utilização das redes neuronais atualmente são, por exemplo, no diagnóstico médico por classificação de imagens, no controlo de qualidade, nos *"chatbots"* automatizados (ex: "ChatGPT"), na organização automática e classificação de dados escritos, no reconhecimento facial utilizado nos smartphones, e também no reconhecimento visual em veículos autónomos [6], que é a aplicação deste trabalho.

2.1.1 *Redes Neuronais como Grafos de Operações*

As redes neuronais são compostas por várias camadas de neurónios, contendo, tal como estes, uma camada de entrada e uma camada de saída, mas também com outras camadas entre as duas enunciadas anteriormente, designadas por camadas ocultas.

O neurónio, que é o elemento base de uma rede neuronal, foi introduzido pela arquitetura *Perceptron*, que é uma arquitetura concebida e documentada na investigação desenvolvida por Frank Rosenblatt [7, 8]. Um neurónio, como se verifica na Figura 2.2, é essencialmente constituído por um conjunto de pesos responsáveis pela ponderação simples do sinal de entrada (operação de multiplicação) e por

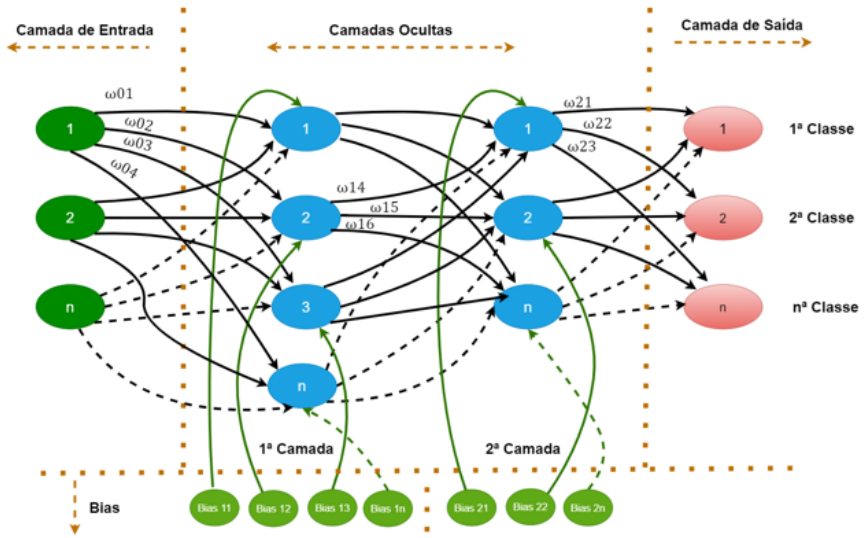

Figura 2.1: Uma simples rede neuronal. Retirada de [6]

um bias (sendo também conhecido como o limiar de ativação) que é uma variável somada ao resultado da ponderação cujo objetivo é deslocar a função melhorando a sua capacidade de ajuste relativamente aos dados fornecidos. O bias também pode permitir que a saída do neurónio seja diferente de zero em casos quando todos os valores de entrada apresentarem um valor nulo.

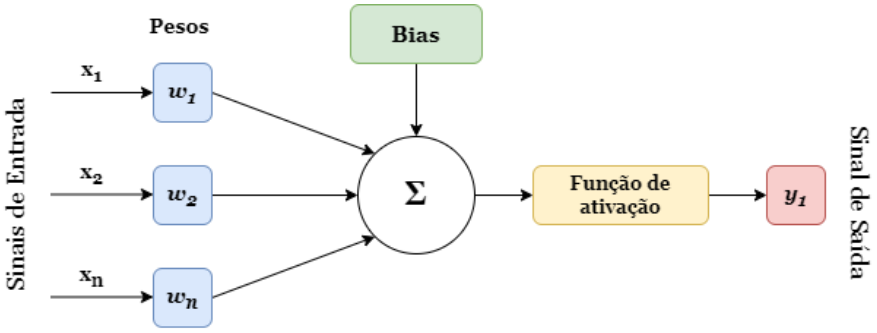

Figura 2.2: Representação do esquema de funcionamento de um neurónio.

Considerando o grafo de operação da Figura 2.1, tal como num neurónio simples, as redes neuronais podem ter um ou vários sinais de entrada, sendo que o seu número vai depender das características existentes num possível conjunto de dados. Os neurónios da camada de entrada vão ligar à primeira camada oculta, sendo que uma camada oculta é constituída por neurónios "intermédios" e numa rede neuronal podem existir uma ou várias camadas desse tipo, dependendo do grau de complexidade da rede. Neste exemplo, a rede possui duas camadas ocultas, ou seja, a 1ª Camada e a 2ª Camada, mas poderá ter até "n" camadas. Seguidamente, os neurónios da última camada oculta irão ligar à camada de saída, sendo que na camada de saída também podemos ter "n" neurónio de saída. O número de

saídas vai depender do número de opções possíveis em relação ao conjunto de dados característico.

2.1.2 *Transformações e Ativações*

Uma das escolhas mais essenciais e influentes que um engenheiro de ML tem de realizar é a escolha da função de ativação que será utilizada nos neurónios da rede neuronal que irá estudar. Essa escolha depende da estrutura, do conjunto de dados e do objetivo que a rede neuronal irá ter. As funções de ativação desempenham um papel fundamental nas redes neuronais profundas, atuando como transformações não lineares que introduzem complexidade e permitem que o modelo aprenda padrões e representações complexas a partir dos dados. Em termos gerais, as funções de ativação são necessárias de modo a evitar a linearidade, já que sem elas, os dados passariam pelos neurónios e pelas camadas da rede neuronal apenas através de funções lineares (como por exemplo $y = ax + b$). A composição dessas funções lineares é novamente uma função linear e, portanto, independentemente do número de camadas pelas quais os dados passam, a saída será sempre um resultado de uma função constante ao longo dos *inputs*, o que poderá levar a uma pobre modelação dos dados para problemas mais complexos [9, 10], como se verifica na Figura 2.3.

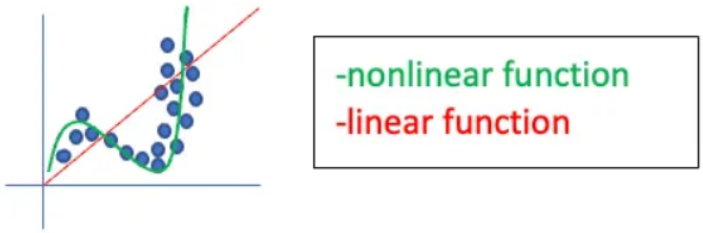

Figura 2.3: Um exemplo que mostra as vantagens das funções não lineares para o ajuste de modelos de dados. Retirada de [10]

Existem muitos tipos de funções de ativação tendo em conta o tipo de rede neuronal, nomeadamente as funções de degrau unitário, as lineares e as não lineares. Dentro de todas estas, as três funções de ativação mais comummente usadas e melhor explicitadas em seguida são a função Unidade Linear Retificada (em inglês, *Rectified Linear Unit* (ReLU)), a função Sigmoide (função Logística) e a função Tangente Hiperbólica (em inglês, *Hyperbolic Tangent* (Tanh)) [6, 9, 10]:

- Unidade Linear Retificada (ReLU)
  A função ReLU é uma função não linear e computacionalmente leve, definida pela equação 2.1. Esta função tornou-se numa das funções mais amplamente utilizadas devido à sua simplicidade e eficácia, dado que substitui todos os valores de entrada negativos por 0, e deixa os valores positivos inalterados

(Figura 2.4). A função ReLU também ajuda a mitigar o problema do gradiente desvanecente e acelera a convergência dos processos de treino da rede neuronal, no entanto, sofre do problema do "ReLU morto", onde os neurónios tendem a se tornar inativos (produzindo o valor 0) indefinidamente durante o processo de treino da rede neuronal.

$$f(x) = \begin{cases} x, \text{ se } x \geqslant 0 \\ 0, \text{ se } x < 0 \end{cases} \tag{2.1}$$

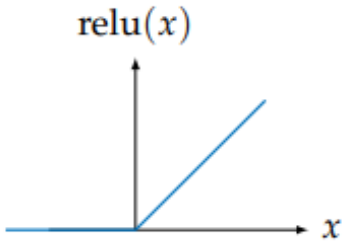

Figura 2.4: Unidade Linear Retificada. Retirada de [9]

- Função Sigmoide
  A função Sigmoide, também conhecida como função Logística, é definida pela equação 2.2. Esta função comprime os valores de entrada produzindo um valor de saída entre 0 e 1 (Figura 2.5), o que lhe torna uma função útil para problemas de classificação binária, pois é possível definir um valor limite de "probabilidade" (sendo daí que provém o nome de função Logística). Acima desse valor, a saída pode ser classificada como 1, e abaixo dele, a saída será 0, sendo que é por essa mesma razão que torna a função Sigmoide uma escolha popular para a camada final de uma rede neuronal. Mas, uma das desvantagens desta função de ativação é que o cálculo de expoentes é complexo e, portanto, pode tornar as redes de maior profundidade/complexidade mais lentas já que como a função não é centrada em torno de 0, então a otimização dos parâmetros das camadas seguintes é dificultada. A função também possui algumas partes que são quase completamente planas/lineares, o que leva a gradientes muito pequenos, sendo que esse facto torna a descida do gradiente extremamente lenta caso a saída seja um valor muito positivo ou um valor muito negativo. A este fenómeno dá-se o nome de saturação e de problema do gradiente desvanecente, como já enunciado na função de ativação ReLU.

$$\sigma(x) \text{ ou } f(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

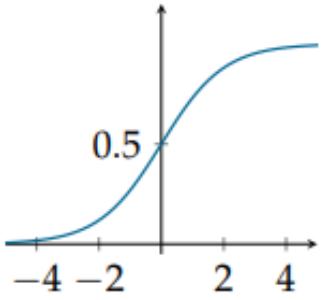

Figura 2.5: Função Sigmoide. Retirada de [9]

- Função Tangente Hiperbólica (Tanh)
  A função Tanh é definida pela equação 2.3. Esta função de ativação, equitativamente com a função Sigmoide, comprime os valores de entrada, mas desta vez entre os valores de saída de -1 e 1 (Figura 2.6). Desta forma, mitiga o problema do gradiente desvanecente em certa medida, já que agora a sua saída varia entre -1 a 1 e a sua função encontra-se centrada em 0 o que facilita a otimização dos parâmetros nas camadas que se lhe seguem, mas ainda sofre de gradientes desvanecentes para valores de dados de entrada extremos.

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.3}$$

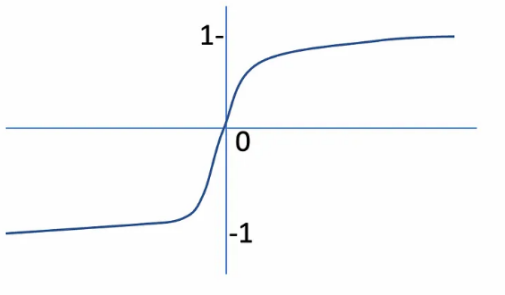

Figura 2.6: Função Tangente Hiperbólica. Retirada de [10]

Para além das três funções de ativação enunciadas, também existem bastante mais funções que são usadas em redes neuronais profundas, mas à altura da realização deste trabalho, com um menor nível de uso. Alguns exemplos são a função *Leaky* ReLU que aborda o problema do "ReLU morto" permitindo uma inclinação pequena e positiva (geralmente definida como uma constante de pequeno valor como 0.01) para valores de entrada negativos, o que garante que os neurónios não se tornam completamente inativos e possam ser recuperados durante o processo de treino da rede; a função Unidade Linear Exponencial (em inglês, *Exponential Linear Unit* (ELU)) que é semelhante à função Leaky ReLU, mas com a adição de uma função exponencial para valores de entrada negativos, o que ajuda a aliviar o problema

do "ReLU morto"; a função Unidade Linear Exponencial Dimensionada (em inglês, *Scaled Exponential Linear Unit* (SELU)) que é uma variação da função ELU com um fator de escala específico e propriedades de normalização, e a função *Softmax* que é mais usualmente utilizada na camada de saída de uma rede neuronal para tarefas de classificação multi-classe [11].

2.1.3 *Diferentes Tipos de Redes Neuronais*

De um modo informal, uma rede neuronal profunda pode ser denotada como um grafo sem sentido algum cheio de nós/neurónios interconectados por setas a apontar para todo o lado. Porém, na prática, as redes neuronais costumam estar organizadas por diferentes camadas (em inglês, *layers*), e por isso existem diferentes tipos de redes pelo facto de que existem diferentes tipos de camadas.

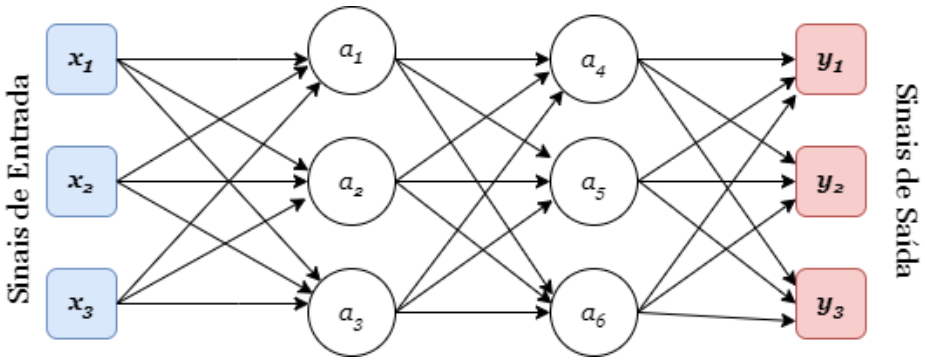

Figura 2.7: Um perceptrão multicamada.

Observando o grafo de operação obtido da Figura 2.7, neste exemplo têm-se três entradas (x_1, x_2 e x_3) e três saídas (y_1, y_2 e y_3), sendo que tendo em atenção todos os neurónios da rede neuronal, também se pode indicar que foram formadas camadas, isto é, a camada de entrada, a camada de saída e duas camadas centrais, que são chamadas de camadas ocultas como já explicitado no subcapítulo 2.1.1. A esta forma de grafo (ou arquitetura) de operação dá-se o nome de perceptrão multi-camada (em inglês, *Multilayer Perceptron* (MLP)) [9]. As camadas ocultas de um MLP (sendo no caso da Figura 2.7, duas camadas) também são chamadas de camadas totalmente conectadas (em inglês, *fully connected layers*) pois cada neurónio dessas duas camadas (a_1, a_2, a_3, a_4, a_5 e a_6) recebe todos os valores dos sinais de saída da camada anterior.

As redes neuronais são normalmente utilizadas como métodos/ferramentas de classificação, ou seja, recebem uma entrada, por exemplo, os pixeis de uma imagem, e classificam o que está na imagem (a classe da imagem). Quando se realiza o processo de classificação, a camada de saída do perceptrão multi-camada representa a probabilidade de cada classe, isto é, por exemplo, a partir da Figura 2.7, y_1 pode

ser representada como a probabilidade de a entrada ser classificada como um sinal de STOP, y_2 pode ser retratada como a probabilidade de a entrada ser classificada como um sinal de trânsito de cedência de passagem e y_3 pode ser simbolizada como a probabilidade de a entrada ser classificada como um sinal de trânsito de limite de velocidade máxima. De forma a garantir que as probabilidades estejam normalizadas, isto é, estejam entre os valores de 0 a 1 e o seu somatório resultar em 1, então a camada final (camada de saída) deve aplicar por exemplo a função de ativação *Softmax* ou Sigmoide [9].

Existem vários tipos de DNNs, sendo que cada uma tem a sua própria função. Mas, as redes neuronais que irão ser abordadas com um maior nível de detalhe em seguida e que são mais habitualmente utilizadas para casos/exemplos em contexto de condução autónoma são as Redes Neuronais *Feedfoward* (em inglês, *Feed-Forward Neural Network* (FFNN)) e as Redes Neuronais Convolucionais (em inglês, *Convolutional Neural Network* (CNN)) [2].

- Redes Neuronais *Fedfoward* (FFNN)
  Uma rede neuronal *feedfoward* é um dos tipos mais simples de redes neuronais existentes. Nesta rede, a informação move-se apenas numa direção (para a frente), a partir dos neurónios de entrada, atravessando os neurónios ocultos (caso existam) até aos neurónios de saída, isto é, nesta rede não há ciclos ou *loops*. As redes neuronais *feedfoward* foram o primeiro tipo de rede neuronal a ser inventado e são mais simples do que as suas congéneres, tal como as redes neuronais recorrentes (em inglês, *Recurrent Neural Network* (RNN)) e as redes neuronais convolucionais [6, 12].

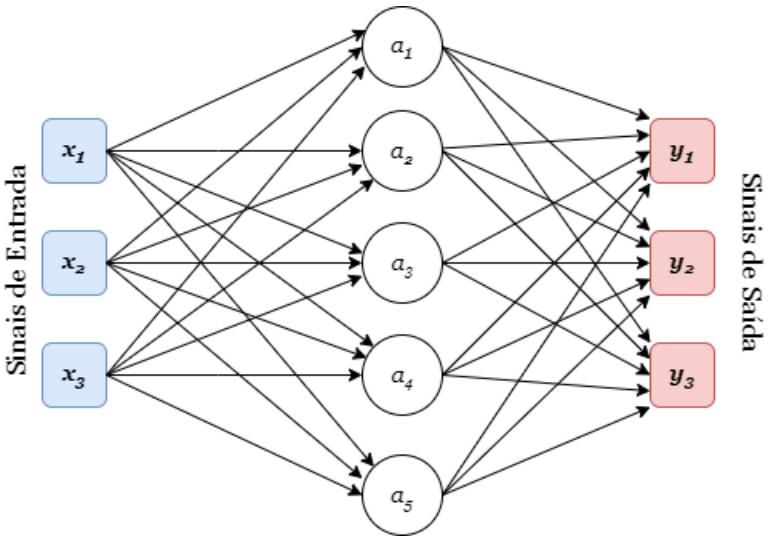

Figura 2.8: Exemplo de uma rede neuronal *feedfoward* totalmente conectada.

Como se pode verificar pela Figura 2.8, a arquitetura de uma rede neuronal *feedfoward* é constituída por três tipos de camadas, ou seja, a camada de

entrada, a(s) camada(s) oculta(s) e a camada de saída, e cada neurónio de uma camada está ligado a todos os neurónios da camada seguinte, o que faz deste tipo de rede também ser chamada de rede totalmente conectada.

O funcionamento de uma rede neuronal *feedfoward* envolve duas fases: a fase *feedfoward* e a fase *backpropagation* [12]. Na fase *feedfoward*, os dados de entrada são inseridos na rede e propagam-se através dela. Em cada camada oculta, a soma ponderada das entradas é calculada e transmitida a uma função de ativação que introduz uma não-linearidade no modelo. Esse processo continua até que a camada de saída seja alcançada e uma previsão da classificação seja efetuada. Relativamente à fase de *backpropagation*, quando uma previsão é realizada, o erro, que é a diferença entre os dados de saída previstos e os dados de saída reais, é calculado. Seguidamente, esse erro é então propagado pela rede, e os pesos são ajustados de forma a minimizar o erro. O processo de ajuste dos pesos dos neurónios é feito normalmente utilizando um algoritmo de otimização de descida do gradiente durante o processo de treino da rede.

A forma de treino envolve a utilização de um conjunto de dados para corrigir os pesos das ligações entre os neurónios. Isso é efetuado através de um processo iterativo em que o conjunto de dados é passado pela rede várias vezes e, de cada vez, os pesos são atualizados de forma a reduzir o erro de previsão. Este processo é conhecido como descida do gradiente e continua até que a rede tenha um desempenho satisfatório nos dados de treino [12].

Embora as redes neuronais *feedfoward* sejam eficazes, elas têm o seu próprio conjunto de desafios e limitações. Um dos principais desafios é a escolha do número de camadas ocultas e do número de neurónios em cada camada, o que pode afetar significativamente o desempenho da rede [12]. Outro problema comum é o *overfitting* [2], onde a rede pode aprender os dados de treino demasiado bem, incluindo o ruído correspondente, e depois possui um desempenho paupérrimo com dados novos, *i.e.* que sejam diferentes dos dados de treino.

Assim, as redes neuronais *feedfoward* são um conceito fundamental no domínio das redes neuronais e do *deep learning* porque proporcionam uma abordagem simples de maneira a modelar os dados e fazer previsões em função desses dados, e também abriram caminho para arquiteturas mais complexas em novas aplicações de AI.

- Redes Neuronais Convolucionais (CNN)
  No domínio do *deep learning*, as CNN são o algoritmo mais famoso e mais utilizado [13]. Têm sido amplamente aplicadas numa série de domínios diferentes, incluindo a classificação de imagens [14], o processamento de voz [13], o reconhecimento facial [6], entre outros, e distinguem-se das outras redes neuronais pelo seu melhor desempenho com diferentes tipos de entradas de sinais,

tais como imagens, voz ou áudio. E no mesmo raciocínio, comparativamente a redes neuronais tradicionais totalmente conectadas, a aplicação de camadas convolucionais permite utilizar um menor número de pesos partilhando o mesmo núcleo (em inglês, *kernel*) ao longo de todo o conjunto de dados de entrada. Desta forma, as redes neuronais convolucionais são computacionalmente menos exigentes, necessitando de um menor armazenamento, pois não necessitam aprender os pesos por cada pixel da amostra, como acontece em camadas totalmente conectadas.

Um modelo de uma CNN é constituído essencialmente por três tipos principais de camadas e duas etapas [6, 13, 14]. Os três tipos principais de camadas que podem existir numa rede neuronal convolucional, que irão ser explicados com um maior detalhe posteriormente no capítulo, são as camadas convolucionais (em inglês, *convolutional layer*), as camadas de extração de características (em inglês, *pooling layer*), e as camadas totalmente conectadas. Para além destas três camadas fundamentais, é ainda bastante comum a utilização do método de normalização em lote (em inglês, *Batch Normalization*) em redes neuronais convolucionais, sendo que esta técnica é representativa de uma própria camada na rede, ou seja, a camada de normalização em lote (em inglês, *batch normalization layer*) e por isso também irá ser explicitada seguidamente no capítulo. No que se refere às etapas, a primeira etapa de uma CNN é a extração de características utilizando uma ou várias camadas convolucionais e a segunda etapa é a etapa de classificação responsável por aprender a função que dá resposta ao problema com base na informação extraída na primeira etapa.

Um processo de aprendizagem em redes neuronais convolucionais pode conter uma ou várias camadas convolucionais. Quanto maior ser o número de camadas convolucionais utilizadas, mais profunda é a rede, o que permite extrair informação mais complexa [13, 14]. A cada camada, a rede neuronal convolucional aumenta a sua complexidade, identificando partes maiores (ou mais específicas) da imagem ou do sinal de entrada. As camadas mais próximas da camada de entrada concentram-se em características mais simples, como cores e arestas, enquanto que à medida que os dados de entrada progridem através das camadas seguintes da rede neuronal convolucional, esta começa a reconhecer elementos ou formas mais complexas do objeto até identificar finalmente a classe do objeto pretendido [14].

A etapa de extração de características é composta essencialmente por blocos convolucionais e blocos de redução/agrupamento de características (*pooling*). Já o classificador (etapa de classificação) é formado por uma camada totalmente conectada que interliga os neurónios da etapa anterior a uma camada mais

densa que é responsável pela classificação final [13, 14]. Um exemplo da arquitetura de uma rede neuronal convolucional para classificação de imagens é ilustrado na Figura 2.9.

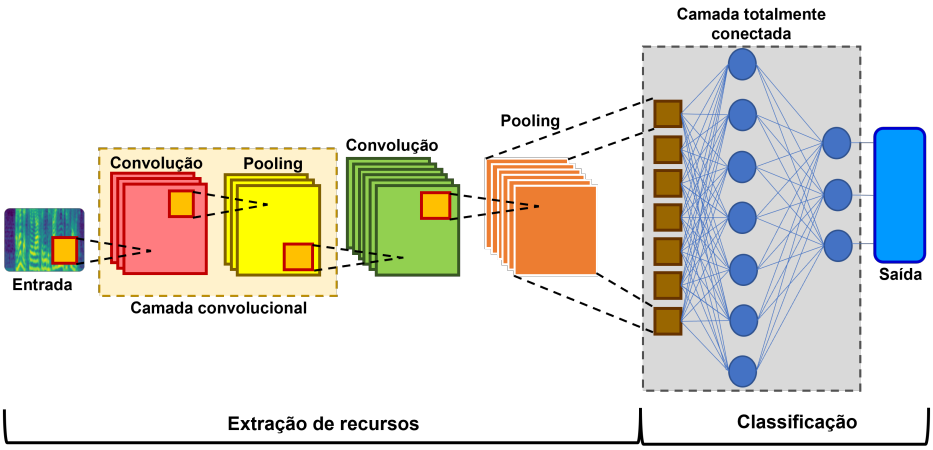

Figura 2.9: Arquitetura de uma rede neuronal convolucional para classificação de imagens, com duas camadas convolucionais.

As camadas que tipicamente compõem as duas etapas de uma rede neuronal convolucional são descritas de seguida com um maior pormenor.

1. Camada Convolucional - A camada convolucional pode-se dizer que é o "bloco de construção central" de uma CNN, porque é onde ocorre a maior parte da computação. Requer alguns componentes, que são os dados de entrada, um filtro e um mapa de características.

   Assumindo-se um exemplo onde a entrada será uma imagem a cores, que é composta por uma matriz de pixeis tridimensional (3D), então isso significa que a entrada terá três dimensões, isto é, a altura, largura e profundidade, que correspondem aos valores de pixeis *Red Green Blue* (RGB) numa imagem. Além disso, têm-se um detetor de características, que também é conhecido como *kernel* ou filtro, que se desloca pelos campos recetivos da imagem, verificando se a característica está presente. A este processo dá-se o nome de convolução [14].

   O detetor de características é uma matriz bidimensional (2D) de pesos que representa parte da imagem de entrada. Embora possam variar em tamanho, o tamanho do filtro é normalmente uma matriz 3x3, 5x5, ou 7x7, sendo que isso determina o tamanho do campo recetivo. O filtro é então aplicado a uma área da imagem e é calculado um produto escalar entre os pixeis de entrada e o filtro. O resultado deste produto escalar é depois introduzido numa matriz de saída, sendo que o filtro desloca-se um passo de cada vez, repetindo o processo até que o *kernel* tenha percorrido toda a imagem, tal como se visualiza na Figura 2.10. A saída final da série

de produtos escalar entre a imagem de entrada e o filtro é conhecida como o mapa de características, ou o mapa de ativação [13,14]. Como se ilustra na Figura 2.10, o *kernel* ou filtro é uma matriz de valores inicialmente aleatórios também designados por pesos. Estes pesos irão ser ajustados ao longo de cada iteração de treino da rede neuronal, aprendendo a extrair características mais significativas e ficando por isso cada vez mais eficazes [13,14].

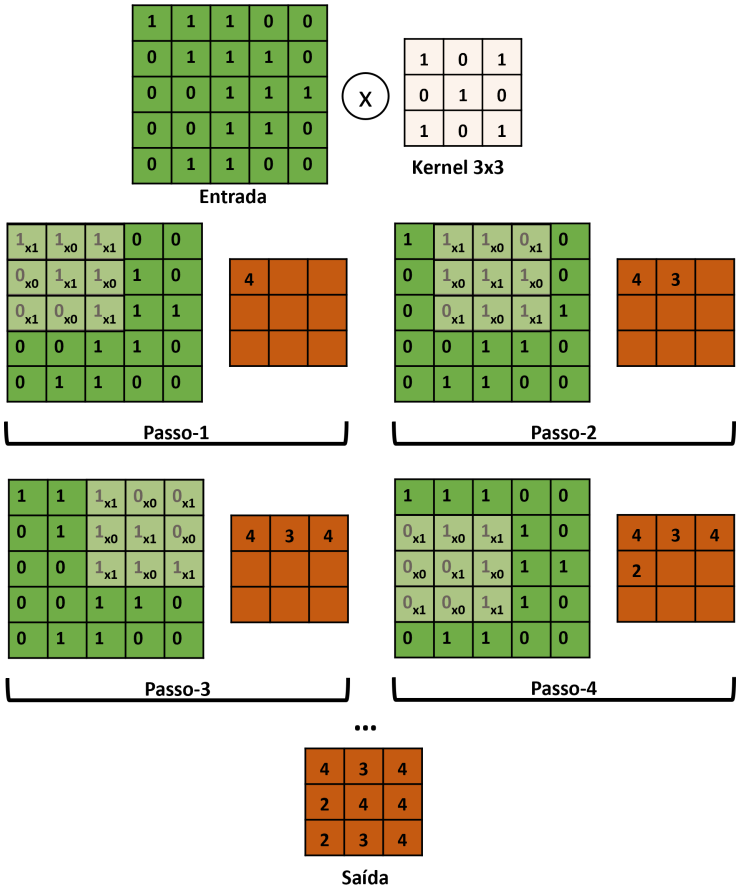

Figura 2.10: Operação de convolução.

Depois da operação de convolução, é tipicamente aplicada uma função de ativação não linear. A função de ativação mais usada nas camadas convolucionais é a ReLU, mas também existem casos onde se utilizam a função Sigmoide, a função Tanh, a Leaky ReLU, entre outras [13]. Logo após a aplicação da função de ativação respetiva, é efetuada a operação de *pooling*.

2. Camada de redução de características (*pooling*) - As camadas de *pooling*, também conhecidas como *downsampling*, efetuam a redução da dimensionalidade, reduzindo o número de parâmetros da entrada. À semelhança da operação de convolução, a operação de *pooling* passa um filtro por todos os parâmetros da entrada, mas a diferença é que o filtro desta vez

não possui nenhum valor de pesos. Em vez disso, o *kernel* aplica uma função de agregação aos valores dentro do campo recetivo, preenchendo assim a matriz de saída [14].

Como é apresentado na Figura 2.11, existem dois tipos de *pooling*, o *Max pooling* e o *Average pooling*. No *Max pooling*, à medida que o filtro se desloca pelos diferentes campos recetivos/parâmetros da entrada, é selecionado o valor de pixel com o valor máximo de forma a ser enviado para a matriz de saída. Enquanto isso, no *Average pooling*, à medida que o filtro se move pelos diferentes campos recetivos/parâmetros da entrada, o valor enviado para a matriz de saída é o cálculo do valor médio de todos os pixeis dentro do campo recetivo respetivo em cada iteração [13, 14].

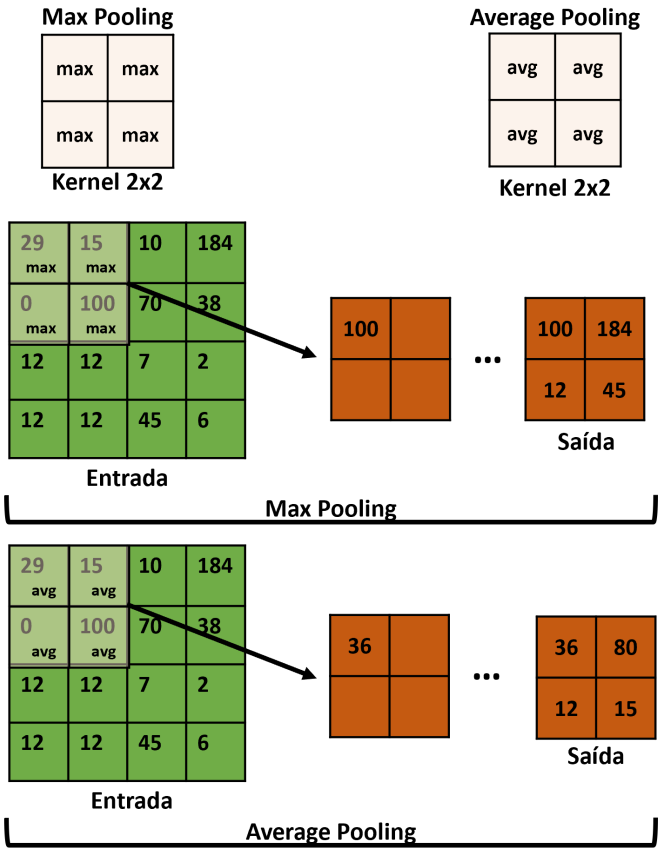

Figura 2.11: Operação de *pooling*.

Embora muita informação seja perdida na camada de redução de características, esta operação também possui uma série de benefícios, já que ajuda a reduzir a complexidade, melhora a eficiência e limita o risco de *overfitting* da rede [14].

3. Camada/Rede totalmente conectada - Normalmente, esta camada está localizada no final de cada arquitetura de redes neuronais convolucionais. No interior desta camada, como já descrito na Figura 2.8, cada neurónio

está conectado a todos os neurónios da camada anterior. Esta camada efetua a tarefa de classificação com base nas características extraídas através das camadas anteriores (camadas convolucionais e de extração de características) e dos seus respetivos filtros. Enquanto as camadas convolucionais e de *pooling* tendem a utilizar a função de ativação ReLU, as camadas totalmente conectadas utilizam geralmente uma função de ativação *Softmax* a fim de classificar os valores de entrada de forma adequada, produzindo os sinais de saída como uma probabilidade entre 0 e 1 [14].

4. Batch Normalization - A Normalização em Lote [15] é uma técnica de normalização comum que desempenha um papel crucial no processo de treino de redes neuronais. Muitas vezes, este método é adicionado em forma de bloco/camada em redes neuronais convolucionais nas tarefas de classificação de imagens. Existem duas linhas de investigação relativamente ao local onde a camada de normalização em lote deve ser colocada na arquitetura convolucional, isto é, antes ou depois da função de ativação. Enquanto o artigo original [15] coloca-a (e aconselha) antes da função de ativação, existem também alguns artigos/trabalhos (em minoria) que aconselham colocar depois [16, 17]. Relativamente ao funcionamento, a normalização em lote é apenas outra camada da rede que é inserida entre uma camada oculta e a camada oculta seguinte. A sua função, tal como se verifica na Figura 2.12, é receber as saídas da primeira camada oculta (por exemplo, os pontos azuis) e normalizá-las (por exemplo, os pontos vermelhos) antes de as transmitir como entrada para a camada oculta seguinte.

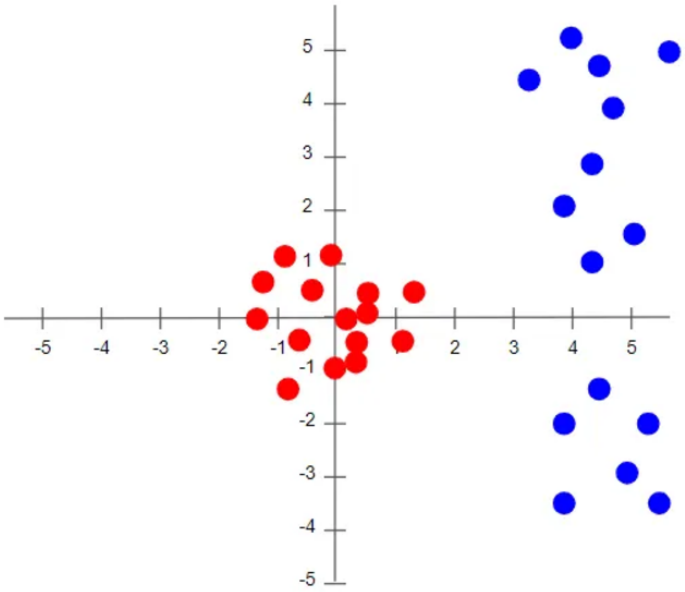

Figura 2.12: Representação de dados normalizados. Os pontos azuis simbolizam os dados não normalizados (antes da camada de normalização em lote), e os pontos vermelhos simbolizam os dados normalizados (depois da camada de normalização em lote). Retirada de [17]

Apesar de não ser muito detalhada neste trabalho, uma classe de redes muito frequentemente utilizadas num contexto de condução autónoma, são as redes neuronais recorrentes (RNN). Uma rede neuronal recorrente é um tipo de rede neuronal que usa dados sequenciais ou dados de séries temporais. Esses algoritmos de *deep learning* são comummente utilizados em problemas ordinais ou temporais, tais como a tradução de linguagem, o processamento de linguagem natural, o reconhecimento de voz e a legenda de imagens. Esta rede neuronal é assim incorporada em aplicações populares do nosso dia-a-dia, como a *Siri* para utilizadores da marca Apple, a pesquisa por voz e o Google Tradutor [6]. Uma característica distinta das redes neuronais recorrentes é que elas partilham parâmetros entre cada camada da rede, ou seja, enquanto as redes neuronais *feedfoward* possuem pesos diferentes em cada neurónio, as redes neuronais recorrentes partilham o mesmo parâmetro de peso dentro de cada camada da rede [6]. Ainda assim, as RNNs tendem a deparar-se com dois problemas, conhecidos como gradientes explosivos e gradientes que desaparecem [6].

2.2 VERIFICAÇÃO FORMAL

As redes neuronais, ou mais especificamente, as redes neuronais profundas são concebidas para serem utilizadas, com entradas arbitrárias. Este facto gera uma grande incerteza na determinação do comportamento de um modelo de rede neuronal, mesmo que este tenha sido pré-treinado [18], muito por causa da instabilidade em relação a perturbações adversárias que as redes neuronais possuem, isto é, alterações

mínimas numa imagem de entrada pode ser a causa de a rede efetuar uma classificação incorreta. Consequentemente, a confiabilidade das redes neuronais tornou-se um novo desafio para as diferentes comunidades de investigação, especialmente para as que são utilizadas em sistemas críticos para a segurança de pessoas e equipamentos, tal como a condução autónoma [19]. Por estas razões, existe uma grande procura de métodos fiáveis e rigorosos de forma a se verificar formalmente a robustez dos modelos de redes neuronais.

Inicialmente, este subcapítulo apresenta a verificação formal numa das suas implementações típicas, denominada de verificação de modelos, incluindo os seus elementos e processos fundamentais, e também uma possível definição de verificação de propriedades, enquadrando a definição de redes neuronais no âmbito da verificação formal. Em seguida, apresenta-se uma introdução sobre a definição da perturbação e do seu modo de medição em perspetiva de redes neuronais, tal como também é formalizada a propriedade de robustez usada durante a realização do trabalho. Logo depois, é explicitado de uma maneira mais percetível o problema da avaliação da correção dos resultados obtidos, e seguidamente são descritos e enunciados os diferentes modelos de redução e raciocínios de especificação que existem para o processo da verificação formal.

2.2.1 *Contexto*

A verificação formal [9, 19, 20] é definida como um método que prova ou refuta matematicamente a fiabilidade de um sistema específico de software ou hardware através de métodos formais. Enquanto os testes tradicionais são normalmente utilizados na procura de falhas/defeitos (*bugs*) através de uma execução ou simulação real, a verificação formal enfatiza o facto de se provar a fiabilidade num contexto de modelos referentes ao sistema examinado no que diz respeito a uma determinada especificação ou propriedade, e, por conseguinte, a verificação formal exige uma compreensão aprofundada do sistema a examinar e tem normalmente um custo mais elevado do que os testes tradicionais [21].

A verificação de modelos é uma técnica típica de verificação formal utilizada para verificar sistematicamente se uma propriedade formalmente definida é valida ou não num modelo de estados finitos [22]. Foi concebida para explorar exaustivamente todos os estados possíveis do sistema e, no final, fornecer um resultado qualitativo que determina se a propriedade especificada é válida no sistema. Na Figura 2.13, é apresentada uma visão esquemática clássica dos diferentes processos que são realizados na verificação formal através da verificação de modelos, que é composta por três componentes principais: a formalização da propriedade, a abordagem de redução e o raciocínio de especificação [19].

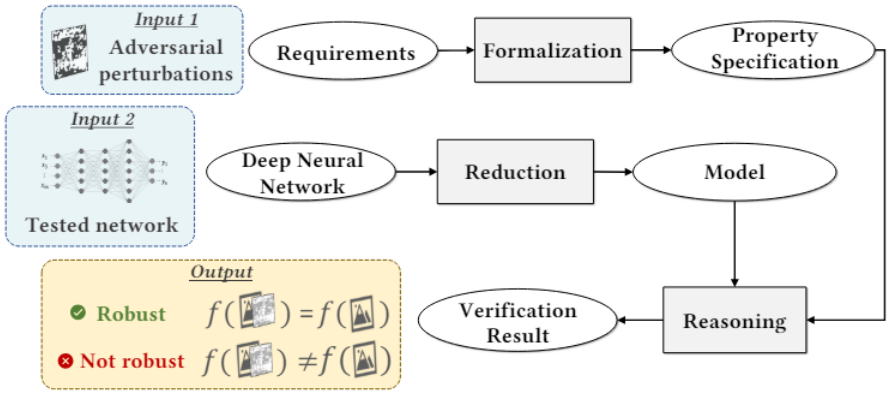

Figura 2.13: Uma visão geral do método de verificação da robustez adversarial de classificação de imagens numa perspetiva de verificação formal. Retirada de [19]

Uma especificação precisa e inequívoca da propriedade a ser formalizada é necessária para o processo de verificação formal. A especificação de propriedades é uma declaração da propriedade, obtida através da formalização dos requisitos do sistema. Esta é frequentemente apresentada sob a forma de uma fórmula lógica clássica destinada a especificar a relação ou a restrição existente entre diferentes componentes do sistema, ou ilustrada numa fórmula lógica temporal a fim de impor uma regra sobre o sistema em termos de tempo. Já a abordagem de redução é uma representação formal do sistema a examinar, que mantém todas as suas características e funções essenciais de forma a garantir uma correta simulação de todos os estados possíveis do sistema. Em suma, a formalização da propriedade define o que se espera que o sistema cumpra, e a modelação do sistema após a redução prescreve a forma como o sistema se comportará. A formalização da propriedade e uma abordagem de redução adequadas constituem a fase de especificação da verificação.

Formalizados o sistema e as propriedades durante a fase de especificação, o verificador executa a tarefa de cálculo/raciocínio e produz um resultado, sendo esta etapa chamada de fase de verificação. Um resultado positivo significa que a especificação da propriedade é satisfeita no modelo examinado, no entanto, um resultado negativo por vezes não deve ser diretamente interpretado como uma violação da especificação da propriedade, pelo que é necessária uma análise adicional. Uma vez que a tarefa de cálculo do verificador produz normalmente um contra-exemplo juntamente com um resultado negativo, então é necessário determinar se o contra-exemplo gerado reflete fielmente a situação do sistema real, isto é, se se trata de um verdadeiro negativo ou um falso negativo. Caso se conclua que se trata de uma falso negativo, então é necessária outra etapa de verificação após o respetivo aperfeiçoamento da abordagem de redução da rede neuronal.

Na verificação de propriedades de redes neuronais profundas através da ótica da verificação formal [19], o modelo de rede neuronal profunda é tratado como uma

função que mapeia uma entrada numa saída, formalizando assim a especificação da propriedade como uma fórmula lógica que envolve tanto a entrada como a saída da rede neuronal.

Tomando como exemplo a propriedade de robustez adversarial e supondo que se têm uma rede neuronal f: $X \rightarrow Y$ que recebe uma entrada x e produz uma saída y como um resultado de classificação, então conjuntamente com um limite de perturbação máxima de entrada ϵ, é possível afirmar que a rede neuronal não é robusta caso seja observado um valor arbitrário (x, y) do conjunto de treino tal que $f(x + \epsilon) \neq y$. Caso contrário, ou seja, caso $f(x + \epsilon) = y$, então prova-se que a propriedade de robustez adversarial é válida na rede neuronal.

Através do exemplo de verificação acima indicado, é possível determinar se a propriedade se mantém no modelo alvo na perspetiva do pior caso possível. Porém, não se tem conhecimento sobre a percentagem de entradas comprometidas em todo o campo de entrada, ou seja, das entradas que são manipuladas com sucesso pelo atacante. Por esta razão, para se efetuar uma verificação mais complexa e aprofundada das propriedades dos modelos de redes neuronais pode ser necessário utilizar um raciocínio quantitativo, sendo o que torna diferente a verificação quantitativa da verificação qualitativa é que na verificação quantitativa é efetuada a contagem de todos os resultados desejáveis obtidos para posterior apresentação de um resultado probabilístico relativo à comparação da respetiva contagem com todos os elementos que são verificados, enquanto que na verificação qualitativa, só é indicado se a propriedade é válida ou não.

Uma vez que este trabalho se concentrará na verificação da propriedade de robustez adversarial, então é assumido que se irá realizar uma verificação qualitativa.

2.2.2 *Formalização da Propriedade*

A maioria das propriedades de uma rede neuronal são propriedades específicas de entrada-saída, isto é, que exigem uma relação específica de mapeamento entre os dados de entrada e os dados de saída. A robustez é uma das primeiras propriedades de relação entrada-saída estudadas que exige que a saída do modelo de rede neuronal seja robusta após pequenas modificações no valor de entrada [19].

Introdução à Perturbação

A perturbação aplicada a uma entrada, como se ilustra na Figura 2.14, pode ser categorizada em dois tipos, nomeadamente a perturbação global e a perturbação regional [23].

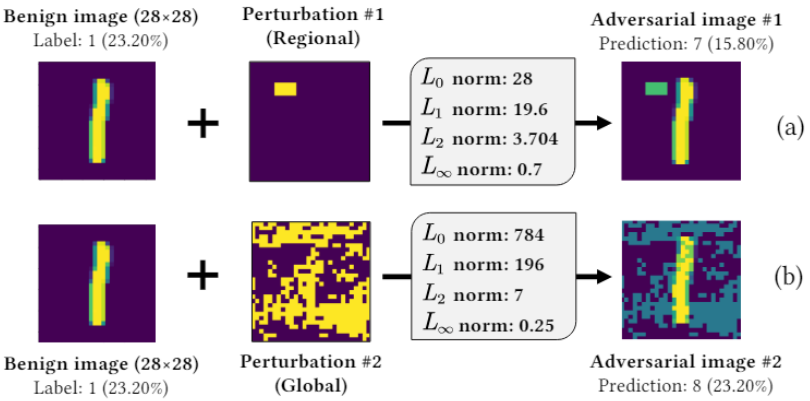

Figura 2.14: Amostras de perturbações adversariais regionais (a) e globais (b) e medição da sua magnitude, ilustradas com ataques à previsão do conjunto de dados MNIST. Retirada e adaptada de [19]

Como o nome indica, a perturbação global é um vetor do mesmo tamanho que as amostras de entrada de uma rede neuronal, e a entrada perturbada é obtida pela sobreposição da perturbação numa entrada benigna. A origem da perturbação global pode ser tanto numa distorção normal existente em implementações reais, como o ruído de um sinal ou as condições climatéricas, como também a resultante de alguns métodos de ataques tal como a descida do gradiente projetado (*Projected Gradient Descent* (PGD)) [20, 23] ou o método do sinal de gradiente rápido (*Fast Gradient Sign Method* (FGSM)) [20, 23]. Já a perturbação regional é geralmente apresentada como um padrão fixo a ser adicionado à entrada original como forma de perturbação, sendo também conhecida como perturbação local ou perturbação localizada. Devido à incerteza do local onde irá ocorrer o ataque, a verificação de robustez de uma rede neuronal contra ataques localizados é mais difícil de se alcançar sem se conhecer o padrão, a localização e a dimensão dos fatores de desencadeamento da perturbação. Além disso, provar a robustez contra um padrão específico de perturbação regional não generaliza a propriedade de robustez contra outro tipo de ataque local. Por estas razões, a perturbação regional é a mais estudada em cenários de verificação de propriedades na perspetiva de redes neuronais porque é a mais difícil de se combater.

Medição da Perturbação

Existem diferentes formas de medir a magnitude (valor) das perturbações aplicadas aos dados de entrada de um modelo de rede neuronal, sendo que no domínio da matemática, uma norma é uma função que calcula a distância de um vetor à origem sendo um valor não negativo. Desta forma, numa perspetiva de verificação, as perturbações são tratadas como vetores e aplica-se a norma para se definir o seu valor em concreto e, correspondentemente, calcular a sua magnitude. Apesar de as perturbações globais e regionais serem criadas com o mesmo objetivo, que é o de maximizar a possibilidade de erro durante o processo de classificação em modelos de redes neuronais, assegurando ao mesmo tempo que são visualmente indistinguíveis,

já os métodos de medição podem ser diferentes dependendo do tipo de perturbação utilizado.

Ao todo, existem 4 normas de medição da perturbação em contexto de verificação de redes neuronais [19, 20, 23].

- A norma L_0 calcula o número de características dos dados de entrada (por exemplo, o número de pixeis de uma imagem) que são afetadas pela perturbação, ou seja, calcula o tamanho da perturbação, o que é útil para descrever o tamanho de uma perturbação regional;
- A norma L_1 (distância *Manhattan*) e a norma L_2 (distância euclidiana) medem a distância que existe entre as entradas benignas e as entradas com perturbação. Uma das maiores diferenças que existe entre as duas normas é a sua forma de cálculo, ou seja, a norma L_1 calcula a soma dos valores absolutos dos elementos do vetor, enquanto a norma L_2 calcula a raiz quadrada da soma de todos os valores ao quadrado dos elementos do vetor;
- Já a norma L_∞ de uma perturbação regista o maior valor/magnitude de perturbação entre todos os seus elementos como um vetor, e por essa razão é que é amplamente utilizada na verificação da propriedade de robustez de diferentes modelos de redes neuronais porque o processo de verificação será então, muito sucintamente, averiguar se o valor da norma L_∞ ultrapassa o limite de perturbação máxima de entrada dado como ϵ.

A Figura 2.14, como já enunciado antes, demonstra um exemplo de diferentes tipos de perturbação dada uma amostra de entrada benigna do conjunto de dados MNIST e adicionalmente ilustra o valor obtido de diferentes normas de medição da perturbação aplicada nas imagens de entrada.

Propriedade de Robustez contra Perturbações Adversariais

A robustez de um sistema informático pode ser descrita como a capacidade de tolerância a falhas que este irá ter ao lidar com entradas erróneas durante a sua execução, enquanto que no domínio do ML, a robustez é geralmente descrita como a capacidade que um modelo tem de produzir de forma consistente uma saída desejável, tendo em conta algumas perturbações (ou distorções) que possam ter sido efetuadas nos dados de entrada [19].

Uma entrada com uma perturbação qualquer que pode alterar a saída de um modelo é chamada de exemplo adversarial (que provêm do inglês, *adversarial example*), portanto, a avaliação da propriedade de robustez de um modelo de uma rede neuronal profunda é definida pela existência desse tal exemplo adversarial. Em análise, a robustez só faz sentido caso o âmbito da perturbação estiver claramente definido e a entrada considerada benigna também estiver corretamente especificada uma vez que a verificação pode não ser capaz de refletir verdadeiramente a robustez

real do modelo em causa se o âmbito da perturbação estiver definido de uma maneira negligenciável, ou seja, o valor da perturbação estiver demasiado pequeno ou demasiado grande. Quanto à entrada benigna do modelo em análise, tanto pode ser uma imagem utilizada para classificação, como um vetor de valores num domínio contínuo.

Em seguida, é formalizada a propriedade de robustez aplicada no modelo de rede neuronal construído ao longo do trabalho:

DEFINIÇÃO 2.1 - Propriedade de Robustez. *Dado um modelo de rede neuronal que mapeia uma entrada benigna x_a até uma saída y_a e, além disso, produz uma saída y_b dada uma entrada adversarial x_b. E seja $||\cdot||_p$ a função para calcular o valor da norma L_p, então a propriedade de robustez ϕ contra qualquer perturbação adversarial no âmbito de ϵ é definida da seguinte forma:*

$$\phi(x_a,y_a,x_b,y_b,\epsilon) = (||x_a - x_b||_p \leqslant \epsilon) \rightarrow (y_a = y_b) \tag{2.4}$$

2.2.3 *Avaliação da Coerência de Resultados*

Em termos intuitivos, um resultado de verificação positivo implica que o modelo de rede neuronal pode sempre evitar violações de uma propriedade específica se o afirmar e, pelo contrário, um resultado de verificação negativo confirma que o modelo não preserva a propriedade de robustez específica, e por isso, em alguns casos, existe a incoerência entre a afirmação e o facto. Desta forma, nesta subsecção discute-se a solidez (do inglês, *soundness*) e a completude (do inglês, *completeness*) dos modelos de redes neuronais para uma avaliação mais abrangente e precisa da coerência de resultados.

Um método de verificação é chamado de sólido se o modelo de rede neuronal viola uma propriedade (por exemplo a propriedade de robustez), e quando o método termina, afirma sempre que a propriedade foi violada. Igualmente, um método de verificação é considerado não sólido se, quando o modelo viola uma propriedade, o método pode potencialmente terminar declarando que a propriedade é satisfeita, isto é, que não foi violada.

Já um método de verificação é chamado de completo caso seja capaz de provar que uma propriedade é válida, sendo essa propriedade efetivamente válida. Semelhantemente, um método de verificação é considerado incompleto caso não consiga garantir que uma propriedade é válida, quando a propriedade é realmente válida [23].

A relação que existe entre a solidez e a completude dos modelos de redes neuronais, encontra-se exemplificada na Tabela 2.15. Nesta tabela encontra-se especificada a

verificação de classificação de imagens relativas a sinais de trânsito, onde o eixo dos x representa a hipótese de resposta (realidade), ou seja, se o resultado é o desejável, ou se existe uma má previsão, e o eixo dos y representa a decisão da rede neuronal (avaliação), isto é, se a propriedade de robustez é válida ou inválida no modelo [24].

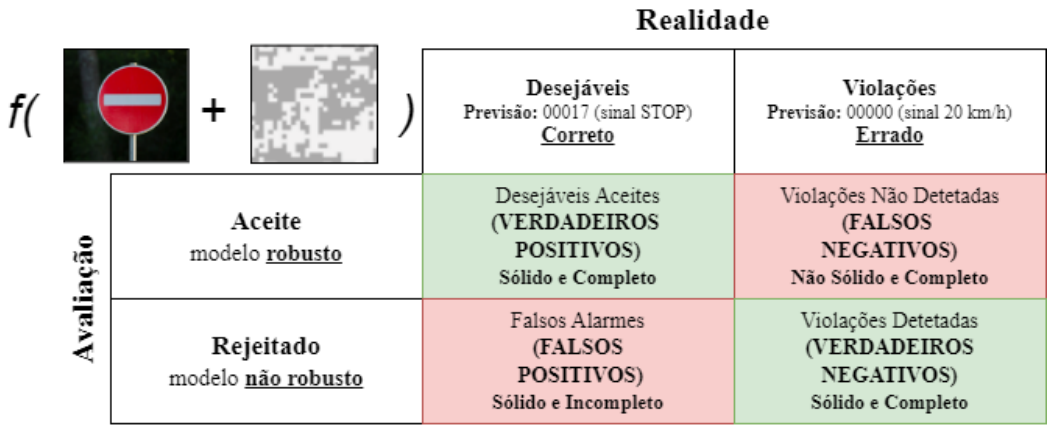

Figura 2.15: Tabela especificada para as hipóteses de verificação de classificação de imagens relativas a sinais de trânsito. Adaptada de [19, 24]

Como se pode constatar a partir da Figura 2.15:

- Uma verificação sólida previne violações não detetadas (falsos negativos), mas pode tolerar falsos alarmes (falsos positivos) no caso de um evento não desejável não se verificar efetivamente, mesmo que o método de verificação indique que a propriedade foi rejeitada. Por outras palavras, o método de verificação pode indicar que o modelo não é robusto, mesmo que o modelo seja verdadeiramente robusto;
- Uma verificação completa evita falsos alarmes (falsos positivos), no entanto é capaz de não contabilizar todas as violações não detetadas (falsos negativos), o que significa que pode existir algum tipo de violação da propriedade não encontrada pelo método de verificação. Neste caso, o método de verificação pode declarar que o modelo é robusto, mesmo que não seja inteiramente verossímil pelo simples facto de não ter encontrado todas as violações da propriedade;
- Uma verificação sólida e completa pressupõe uma previsão perfeita, isto é, que apenas produza resultados desejáveis aceites (verdadeiros positivos) e resultados com violações detetadas (verdadeiros negativos);
- Nas duas categorias assinaladas a verde, pode-se dizer que a avaliação acerta em cheio na realidade, ou seja, os resultados desejáveis aceites são corretamente aprovados, e as violações detetadas são corretamente rejeitadas;

- E, nas outras duas categorias assinaladas a vermelho, a avaliação está errada, porque as violações não detetadas são erradamente transmitidas, e os falsos alarmes são erradamente aceites.

Para se obter uma previsão perfeita de classificação de imagens num modelo de rede neuronal é necessária uma verificação sólida e completa, no entanto, no domínio da AI, a implementação de uma verificação sólida, completa e que termine num intervalo razoável de tempo, continua a ser desafiante. Na prática, um destes três requisitos é normalmente colocado de parte de modo a garantir os outros dois.

Algumas abordagens optam por não considerar com tanto ênfase a variável do tempo de conclusão na verificação de redes neuronais, em comparação com a solidez e a completude [19], tipicamente esta abordagem aplica-se apenas na verificação de redes neuronais consideravelmente pequenas pois os métodos têm pouca escalabilidade, *i.e.* não são capazes de calcular a completude e a solidez em tempo útil para redes com maior complexidade. No caso de redes neuronais profundas, a escalabilidade torna-se um fator crítico no método de verificação, e assim, considerando que a solidez é mais importante no processo de verificação de modelos, então os métodos de verificação mais recentes, como por exemplo os baseados na abstração (por exemplo o método *Box* [23], *Zonotope* [23], etc), geralmente optam por sacrificar a completude para garantir a solidez e a escalabilidade. É por essa razão que os métodos *Box* e *Zonotope* são considerados incompletos, mas escaláveis.

2.2.4 *Redução do Modelo e Raciocínio de Especificação*

Um dos problemas mais difíceis de se resolver no âmbito da Inteligência Artificial (AI) é a verificação formal de propriedades de redes neuronais profundas, sendo por isso muitas vezes chamado de problema *NP-hard*, ou seja, um problema muito complexo. Esta secção aborda a forma de reduzir a verificação das propriedades de um modelo de rede neuronal a um outro problema com estados finitos e uma solução possível de resultados.

Genericamente, o problema da verificação de propriedades de redes neuronais pode ser reduzido a dois grandes grupos de métodos de redução, isto é, o problema de otimização e o problema de alcançabilidade [19], sendo que a forma como este problema é reduzido a cada grupo específico determina a abordagem, algoritmo, ou raciocínio:

- Problema de otimização - O problema de otimização na verificação das propriedades de redes neuronais consiste em formular a tarefa de verificação como uma tarefa de otimização. Um exemplo consiste no aumento do valor de robustez de uma rede, ao minimizar o efeito da função de perda dos ataques

adversariais que são aplicados no modelo de rede neuronal. Esta abordagem utiliza técnicas de otimização, assegurando soluções exatas e rigorosas, de modo a provar que uma determinada propriedade é ou não válida, ou para encontrar um contra-exemplo que não cumpre a propriedade [19, 20];

- Problema de alcançabilidade - O problema de alcançabilidade na verificação de propriedades centra-se numa interpretação (aproximação) simbólica e geométrica das especificações de entradas e saídas válidas, isto é, na análise do conjunto de possíveis resultados que uma rede neuronal consegue produzir dado um conjunto de entradas. Como se verifica na Figura 2.16, os dados ou conjuntos de entrada concretos x, depois de passados por uma rede neuronal F, resultam no conjunto de saída $F(x)$, mas esse conjunto $F(x)$ é de difícil representação. Por isso, os dados ou conjuntos de entrada x são aproximados semanticamente a uma zona/região z.

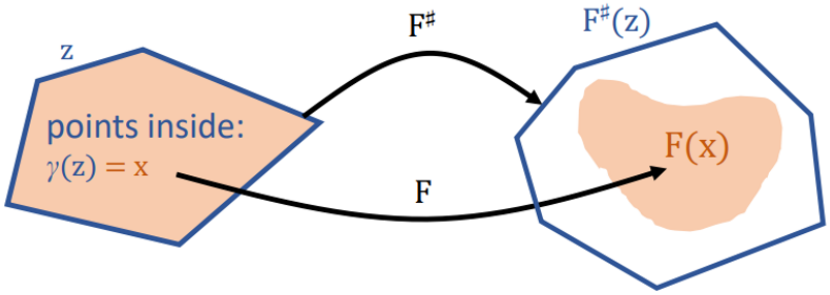

Figura 2.16: Representação da interpretação abstrata dos dados para um problema de alcançabilidade. A região em laranja simboliza os dados concretos e as formas geométricas a azul representam a interpretação abstrata. Retirada de [23]

A aproximação geométrica z, como se observa na Figura 2.17, será propagada camada a camada e refinada em cada operação diferente da rede (por exemplo, convolução, função de ativação ReLU, etc), o que resultará na interpretação abstrata ou sobre-aproximação do conjunto ou dados de saída $F^{\#}(z)$.

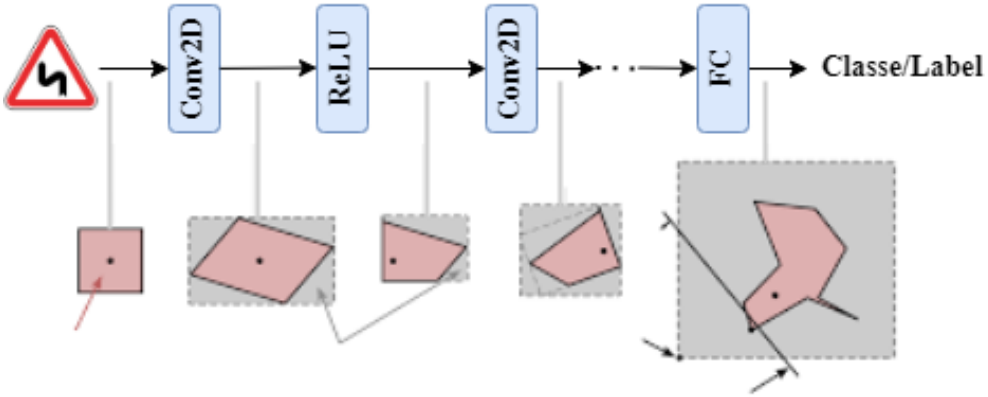

Figura 2.17: Exemplificação da propagação e refinamento camada a camada da aproximação abstrata dos dados de entrada de um sinal de trânsito.

Deste modo, a questão do problema de alcançabilidade na verificação de propriedades é saber se existe um conjunto válido de entradas que conduza a um conjunto válido de saídas, ou seja, se o conjunto de resultados/saídas válidas é alcançável a partir do conjunto de entradas válidas [19, 20, 25].

Dado um modelo de rede neuronal profunda $\mathcal{N}$ e uma propriedade C, a verificação é considerada como um conjunto de técnicas que verifica se a propriedade C é válida no modelo $\mathcal{N}$. Ao contrário de outras técnicas como os ataques e defesas adversariais, uma técnica de verificação necessita de fornecer garantias prováveis de forma a certificar e comprovar os seus resultados. Atualmente, existem três grandes tipos de garantias que validam as diferentes técnicas de verificação [20]:

- Garantia booleana - Quando se utiliza uma garantia booleana, significa que a técnica de verificação utilizada é capaz de afirmar de forma exata e clara quando a propriedade é válida, ou então devolve um contra-exemplo onde a propriedade é inválida;
- Garantia de aproximação - Uma garantia aproximada fornece uma sobre-aproximação ou sub-aproximação (convergência) aos valores dos limites dos conjuntos de saída de alcançabilidade de um modelo de rede neuronal referente a uma propriedade;
- Garantia estatística - Todas as garantias acima referidas necessitam de ser suportadas por uma certificação matemática, mas quando uma certificação matemática é difícil de se obter, uma garantia estatística fornece um limite quantitativo de tolerância ao erro relativamente à afirmação resultante de a propriedade ser válida, ou não. Por outras palavras, a garantia estatística quantifica a probabilidade que a propriedade C possui de ser verdadeira ou válida.

Na Figura 2.18, são apresentadas e classificadas as principais técnicas de verificação de propriedades das redes neuronais existentes na atualidade, focando principalmente nas duas categorias explicitadas anteriormente, *i.e.* o tipo de garantia utilizada durante a verificação em função dos dois grandes grupos de redução do problema:

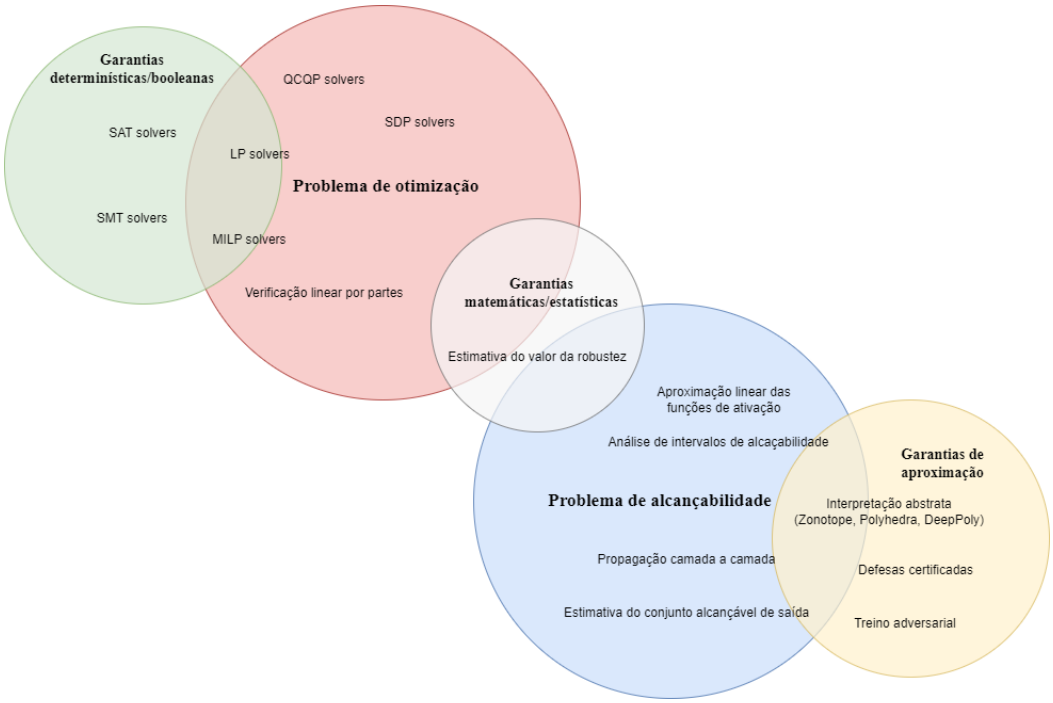

Figura 2.18: Taxonomia das abordagens de verificação de propriedades das redes neuronais. A classificação baseia-se nos principais métodos estudados, dando-se mais ênfase nos de alcançabilidade, de modo a resolver o problema da verificação. Nota: a separação entre a representação geométrica (círculo) de cada abordagem não é rigorosa.

A partir da Figura 2.18, é possível observar que:

- As garantias booleanas, ou também chamadas de garantias determinísticas são maioritariamente usadas no problema de verificação como um problema de otimização. As garantias determinísticas/booleanas são obtidas através da transformação de um problema de verificação num conjunto de restrições, de modo a poderem ser abordadas com um *solver* de restrições. A designação "determinística" ou "booleana" advém do facto de os *solvers* devolverem frequentemente uma resposta determinística ou booleana (exata) a uma questão realizada, ou seja, satisfaz ou não satisfaz. Esta afirmação tem como base o sucesso atual de vários *solvers* de restrições, como os *solvers Boolean Satisfiability* (SAT), *solvers* de *Linear Programming* (LP), *solvers Mixed Integer Linear Programming* (MILP), *solvers Satisfiability Modulo Theories* (SMT), *solvers Quadratically Constrained Quadratic Program* (QCQP) e os *solvers Semidefinite Programming* (SDP) [19, 20]. A verificação linear por partes é uma verificação que pode utilizar 1 ou mais *solvers* dos referidos previamente;
- As garantias de aproximação, como seria de esperar, são predominantemente utilizadas no problema de verificação como um problema de alcançabilidade. Como esta técnica de verificação assenta-se no cálculo de limites superiores e inferiores de intervalos e na propagação desses intervalos por cada camada do modelo de rede neuronal utilizado, então consegue garantir a solidez do

resultado do conjunto de saída, mas não a sua completude. Alguns exemplos destes métodos de sobre-aproximação são a interpretação abstrata, como por exemplo os métodos *Zonotope*, *Polyhedra* e *DeepPoly*, a análise dos intervalos de alcançabilidade, como por exemplo a estimativa do conjunto alcançável de saída em cada camada, e a aproximação linear das funções de ativação não lineares a formas simbólicas [19, 20];

- As garantias estatísticas, ou também chamadas de garantias matemáticas podem ser utilizadas em ambas as reduções do problema de verificação, porque, por exemplo, é possível afirmar a satisfatoriedade de uma propriedade, ou que o valor de um limite superior de um conjunto alcançável é o esperado, na presença de uma certa probabilidade [20].

2.3 ATAQUES ADVERSARIAIS E DEFESAS

As redes neuronais, apesar das suas capacidades notáveis e do seu papel crucial no domínio da Inteligência Artificial (AI), são vulneráveis aos chamados ataques adversariais, e por isso, os ataques adversariais representam às redes neuronais um grande obstáculo relativamente à certificação de robustez e segurança dos seus modelos. Estes ataques envolvem a criação de pequenas perturbações, que são muitas vezes impercetíveis ao olho humano, estrategicamente elaboradas nos dados de entrada de modo a resultar em previsões e saídas incorretas. Desta forma, compreender a natureza destes ataques e o desenvolvimento de defesas robustas com vista a combatê-los é fundamental, especialmente para aplicações em áreas críticas em termos de segurança como a condução autónoma.

Neste contexto, este subcapítulo expõe as duas formas de categorização dos ataques adversariais, isto é, os ataques podem ser categorizados com base no conhecimento do atacante sobre o modelo - ataques *Black-Box* e *White-Box* - e nos objetivos do atacante - ataques direcionados e não direcionados. Seguidamente, apresenta-se uma comparação e um sumário em relação aos diferentes tipos de ataques evidenciados anteriormente, e por fim explicita-se o treino adversarial entre as várias estratégias de defesa existentes, destacando-o como um dos métodos mais eficazes e amplamente investigados.

2.3.1 *Tipos de Ataques Adversariais*

Começando com os ataques com base na quantidade de informações que o atacante (ou agente adversarial) possui sobre o alvo, neste tipo de ataque, o agente adversarial tenta alimentar a rede neuronal com dados de entrada perturbados para que o

classificador os classifique incorretamente ou, em outros casos, tenta apenas recolher informações sobre o modelo. Como já mencionado anteriormente, este tipo de ataque pode ser separado em duas grandes categorias, os ataques *White-Box* e os ataques *Black-Box* [26]:

Ataques *White-Box*

Nos ataques *White-Box*, o atacante dispõe de todas as informações sobre o classificador ou modelo alvo. Estas informações podem ser a base (ou conjunto) de dados de treino, a distribuição dos dados de treino, o tipo de rede neuronal do classificador, as configurações das diferentes camadas, a arquitetura do modelo, etc. Normalmente, num cenário de ataque *White-Box*, o agente adversarial utiliza ataques que se baseiam no gradiente da função de perda da rede neuronal. Já os ataques *White-Box* mais sofisticados utilizam a técnica da retropropagação (do inglês, *backpropagation*) com o objetivo de calcular o gradiente para depois aí criar o ataque, e por isso, essa transparência total permite a formação de ataques altamente complexos e precisos. Os três principais ataques *White-Box* baseados no cálculo do gradiente são:

- *Fast Gradient Sign Method (FGSM)*: O *Fast Gradient Sign Method*, inicialmente introduzido por *Ian Goodfellow* e os seus colaboradores, é uma das primeiras técnicas de ataque [27]. O FGSM utiliza os gradientes de uma rede neuronal de modo a construir uma imagem (ou dados de entrada) adversa. Essencialmente, esta abordagem gera uma imagem adversarial calculando os gradientes de uma função de perda (como o erro quadrático médio ou a entropia categórica cruzada) relativamente à imagem de entrada e, em seguida, utilizando o sinal desses gradientes, cria uma nova imagem, que é formalmente conhecida como imagem adversarial, que maximiza a perda. O resultado é então uma imagem de saída que, para o olho humano, parece idêntica à original, mas provoca o modelo de rede neuronal prever algo diferente do resultado verdadeiro.

  O ataque FGSM pode ser representado como na Equação 2.5 [26].

$$x' = x + \epsilon \cdot sign(\nabla_x \mathcal{L}(\theta, x, y)) \tag{2.5}$$

  Na Equação 2.5, x' é a imagem adversarial criada como saída, x é a imagem original como entrada, y é a categoria da classe real da imagem de entrada (ou seja, o *label* correto), ϵ é uma pequena constante que é multiplicada pelo sinal do vetor do gradiente (isto é, é a intensidade do ruído expressa como um pequeno valor pelo qual os gradientes são multiplicados de forma a criar perturbações), θ é o modelo de rede neuronal utilizado como classificador de imagens e $\mathcal{L}$ é a função da perda.

A Figura 2.19 ilustra uma imagem com uma perturbação, que foi gerada utilizando o ataque FGSM, que, embora seja quase indetetável ao olho humano, leva ao modelo de rede neuronal ter um maior nível de confiança numa saída que contém o *label* incorreto.

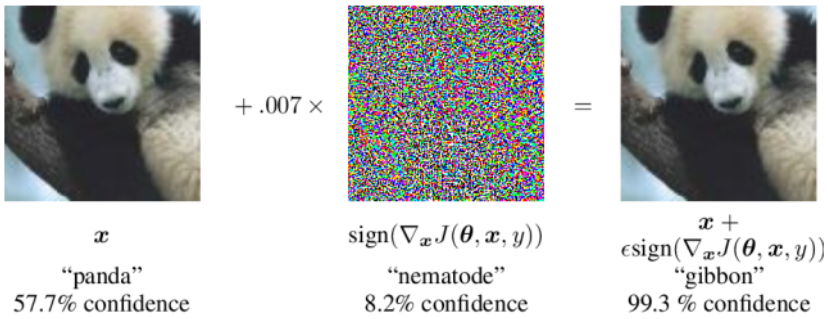

Figura 2.19: Um exemplo de um ataque adversarial utilizando o ataque *Fast Gradient Sign Method*. Retirada de [27]

O ataque FGSM aplicado a uma imagem consiste resumidamente em três passos sequenciais. Primeiro, o valor da função de perda é calculado seguindo uma propagação progressiva dentro da rede, ou seja, começando na camada de entrada e finalizando na camada de saída. Em seguida, são calculados os gradientes relativos aos pixeis da imagem de entrada. E finalmente, os pixeis da imagem de entrada são ajustados de uma forma subtil na direção do sinal dos gradientes calculados para assim maximizar o valor da função de perda [28];

- *Basic Iterative Method (Basic Iterative Method (BIM))*: O *Basic Iterative Method*, que é um método proposto por *Alexey Kurakin*, *Ian Goodfellow* e *Samy Bengio*, é também muitas vezes chamado de *Iterative Fast Gradient Sign Method* (I-FGSM) [29]. A razão para o qual isso acontece é o facto de esta abordagem aplicar o ataque FGSM durante várias iterações com um pequeno incremento (*step size*) entre elas, e também utiliza uma norma máxima a fim de limitar a perturbação adicionada a uma possível amostra de imagem. As imagens adversariais geradas por este ataque podem ser formuladas como se indica na Equação 2.6 [26]:

$$x'_{i+1} = Clip_{x,\epsilon}\{x'_i + \alpha \cdot sign(\nabla_x \mathcal{L}(x'_i, y))\} \text{ onde } x'_0 = x \tag{2.6}$$

Na Equação 2.6, x é a imagem original de entrada, i é uma iteração do ataque, α é uma pequena variável que estipula o tamanho de cada incremento aplicado no ataque até se chegar ao valor de iterações pretendido, $\mathcal{L}(x'_i, y)$ é a função de perda do modelo desejado e $Clip\{\cdot\}$ é a norma que restringe a perturbação adicionada à imagem original;

- *Projected Gradient Descent (PGD)*: O *Projected Gradient Descent*, é uma abordagem de ataque *White-Box* relativamente moderna, desenvolvida por *Madry et al.* com o objetivo de aumentar a robustez adversarial das redes neuronais e desenvolver métodos capazes de treinar redes neuronais robustas a ataques adversariais [30]. Fundamentalmente, o ataque proposto visa encontrar a perturbação que maximiza a perda de um modelo de rede neuronal numa determinada entrada, mantendo o tamanho da perturbação menor do que uma quantidade especificada referida como ϵ. Esta restrição é normalmente expressa a partir das normas L_2 ou L_∞ de perturbação, e é adicionada de modo a que os dados do exemplo adversarial não sejam muito dispares do que os da amostra não perturbada, fazendo com que o exemplo adversarial seja impercetível ao olho humano.

  Deste modo, o PGD é essencialmente uma variante de várias iterações do ataque FGSM e pode ser formulado através da Equação 2.7 [30].

$$x^{t+1} = \Pi_{x+\mathcal{S}}\{x^t + \epsilon \cdot sign(\nabla_x \mathcal{L}(\theta,x,y))\} \tag{2.7}$$

  Embora o ataque PGD possa parecer semelhante ao ataque BIM, o ataque PGD, ao contrário do BIM, inicializa a procura de uma perturbação num ponto aleatório numa área L_∞ por exemplo, o que permite desta maneira, uma procura mais ampla de amostras na área da função de perda. Uma representação visual muito simplista do algoritmo PGD pode ser visualizada na Figura 2.20.

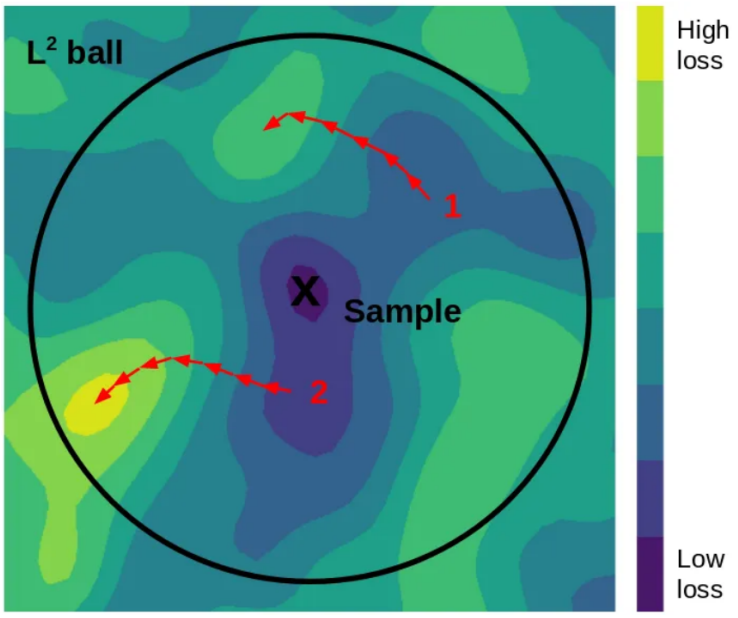

Figura 2.20: O processo iterativo de procura da melhor amostra adversarial do ataque *Projected Gradient Descent*. Retirada de [31]

Através da Figura 2.20, é possível constatar que o processo começa com uma perturbação aleatória na esfera/área L_∞ (ou L_2) em torno de uma amostra, e seguidamente, efetua-se um passo de gradiente na direção da maior perda possível, sendo que caso a perturbação ultrapasse a área da esfera, então o valor da perturbação adquirido é projetado de volta para dentro da esfera. Dando-se os dois exemplos que estão representados, numa primeira tentativa, a amostra adversarial selecionada tem um valor de perda baixo, o que significa que será um exemplo adversarial fraco, mas na segunda tentativa, como a amostra selecionada encontra-se numa região com valores de perdas elevadas, então irá resultar num forte exemplo adversarial [26, 31].

Ataques *Black-Box*

Nos ataques *Black-Box*, o agente adversarial geralmente não tem conhecimento, ou tem muito pouco conhecimento sobre o classificador alvo. Na maioria dos casos, os exemplos adversariais são cuidadosamente construídos de forma a serem enviados para a entrada dos modelos de redes neuronais e o ataque adversarial é construído de acordo com os valores de saída, ou seja, ao contrário dos ataques *White-Box*, os ataques *Black-Box* baseiam-se na observação dos resultados do modelo de modo a inferir as informações necessárias à elaboração de exemplos adversariais. Apesar de o primeiro ataque *White-Box* (FGSM) ter sido criado em 2014, o primeiro ataque *Black-Box* só foi desenvolvido três anos depois (2017), e a maioria dos ataques *Black-Box* só começaram a serem elaborados nos três anos seguintes (2018, 2019 e 2020), por isso é de se notar que a maior parte dos ataques *Black-Box* são mais complexos que os ataques *White-Box*. Apesar disso, um dos ataques *Black-Box* mais simples que foi concebido é o chamado ataque de um pixel:

- Ataque de um pixel: O ataque de um pixel (do inglês, *One Pixel Attack*) foi delineado pelos autores *Su et al.*, que sugeriram uma nova família de ataques *semi-Black-Box*, onde apenas eram modificados um número muito limitado de pixeis de uma imagem de modo a se criar uma amostra adversarial [32]. Contrariamente aos ataques *White-Box*, os requisitos para este ataque são unicamente as classes de saída e as probabilidades de previsão do modelo de rede neuronal alvo. E também, opostamente a outros métodos de ataques, o ataque de um pixel não limita a força do ataque, isto é, não restringe a força da modificação, mas concentra-se em produzir a máxima perturbação possível num único pixel. Por conseguinte, em vez de utilizar uma abordagem baseada no cálculo do gradiente, como os três ataques explicitados no campo dos ataques *White-Box*, este ataque utiliza o algoritmo de evolução diferencial de forma a obter a solução pretendida que, neste caso, é o pixel a ser alterado. A utilização deste ataque possui algumas vantagens, tais como a eficácia, porque é comprovado que possui um elevado grau de desempenho em diferentes bases

de dados, e também a flexibilidade, porque consegue atacar um maior número de tipos diferentes de redes neuronais do que os ataques que são aplicados em redes onde o cálculo do gradiente é difícil [26].

Um exemplo do ataque de um pixel aplicado a quatro imagens da base de dados *ImageNet* é apresentado na Figura 2.21. Como se pode observar, os pixeis que foram modificados estão realçados com círculos vermelhos, e enquanto os *labels* da classe original da imagem estão anunciados a preto, os *labels* da classe que foi prevista pelo modelo e o seu nível de confiança correspondente estão indicadas abaixo.

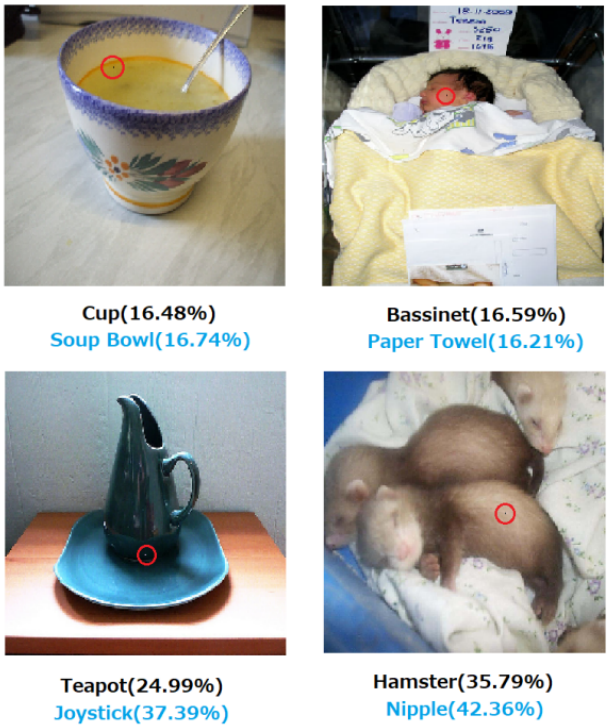

Figura 2.21: Exemplos do ataque de um pixel aplicado a quatro imagens da base de dados *ImageNet*. Retirada de [32]

Seguindo-se para os ataques com base nos objetivos do atacante, neste tipo de ataque, o agente adversarial tenta alimentar a rede neuronal com dados de entrada modificados de forma a que o modelo de rede neuronal os classifique incorretamente ou, os classifique como uma saída específica. Conforme já mencionado previamente, este tipo de ataque pode ser separado em dois grandes grupos, os ataques direcionados (que provêm do inglês, *targeted attacks*) e os ataques não direcionados (que provêm do inglês, *untargeted attacks*) [23]:

Ataques Direcionados

Nos ataques direcionados, o objetivo do atacante é induzir o modelo de rede neuronal em erro de maneira a classificar a entrada (por exemplo, uma imagem) como uma classe de saída (ou *label*) alvo, que seja diferente da classe original e verdadeira. Para tal acontecer, é necessário desenvolver perturbações que não só causam erros

de classificação, como também é fundamental que o façam de uma forma controlada de modo a atingir um resultado (ou *label*) específico.

Uma possível formulação do problema dos ataques direcionados encontra-se apresentada na Equação 2.8 [23].

$$f(x+\eta) = t \tag{2.8}$$

Na Equação 2.8, f: $X \rightarrow C$ representa o modelo de rede neuronal (classificador), $x \in X$ representa os dados de entrada (por exemplo, uma imagem), $t \in C$ representa o *label* alvo (ou classe de saída específica), η representa a perturbação adicionada aos dados de entrada, e $x' = x + \eta$ é o exemplo adversarial gerado.

Dois exemplos onde é aplicado este tipo de ataque são os ataques direcionados *White-Box* FGSM e PGD. Estes dois ataques são versões modificadas dos ataques FGSM e PGD apresentados anteriormente, que visam maximizar a probabilidade de classificação de uma classe de saída particular, em vez de apenas originar uma classificação incorreta no modelo relativamente a uma imagem de entrada.

Ataques Não Direcionados

Nos ataques não direcionados, o objetivo é induzir a rede neuronal a efetuar uma qualquer previsão incorreta, independentemente de alguma classe de saída específica. O foco deste ataque é simplesmente alterar a saída do modelo para um outro *label* que não o correto.

Uma possível formulação do problema dos ataques não direcionados está demonstrada na Equação 2.9 [23].

$$f(x+\eta) \neq f(x) \tag{2.9}$$

Tal como no caso dos ataques direcionados, na Equação 2.9, f: $X \rightarrow C$ representa o modelo de rede neuronal (classificador), $x \in X$ representa os dados de entrada (por exemplo, uma imagem), η representa a perturbação adicionada aos dados de entrada, e $x' = x + \eta$ é o exemplo adversarial gerado, sendo que a única diferença é a não utilização de um *label* alvo como saída, ou seja, neste ataque o importante é ocorrer uma qualquer classificação errada.

Três exemplos onde é aplicado este tipo de ataque são os ataques não direcionados *White-Box* FGSM e PGD, e o ataque não direcionado *Black-Box* de um pixel. Os três ataques tem diferentes técnicas para chegar a uma má classificação, mas ambos possuem o mesmo objetivo, isto é, visam maximizar a função de perda de modo a afastar a maior probabilidade de classificação do modelo de rede neuronal da classe de saída correta.

2.3.2 *Comparação e Sumário dos Ataques Adversariais*

A distinção entre os ataques *White-Box* e *Black-Box* reside principalmente no conhecimento e na metodologia do atacante. Os ataques *White-Box* tiram partido do acesso total que possuem do modelo de rede neuronal, permitindo assim a geração de exemplos adversariais precisos e eficazes. Em contrapartida, os ataques *Black-Box* têm de extrair informações úteis através de meios indiretos, exigindo frequentemente um maior número de análises do modelo e estratégias sofisticadas de modo a obter um sucesso semelhante aos ataques *White-Box*.

A diferença entre os ataques direcionados e não direcionados centra-se no objetivo do atacante. Os ataques direcionados são mais desafiantes, uma vez que exigem que o exemplo adversarial não só consiga enganar a rede, mas também que seja capaz de o classificar numa classe incorreta específica. Já os ataques não direcionados, como são menos restringidos, então têm geralmente taxas de sucesso mais elevadas, uma vez que o seu intuito é apenas provocar um erro de classificação.

Compreender as nuances entre os ataques *White-Box* e *Black-Box*, bem como dos ataques direcionados e não direcionados, é essencial para desenvolver defesas robustas contra exemplos adversariais. Enquanto os ataques *White-Box* fornecem um trajeto claro e óbvio para a criação de exemplos advesariais, os ataques *Black-Box* demonstram a plausibilidade dos ataques no mundo real em que os detalhes do modelo são desconhecidos. Ora, a escolha entre os ataques direcionados e não direcionados influencia ainda mais a complexidade e a taxa de sucesso das estratégias adversariais.

Na Tabela 2.1 são resumidos e comparados os principais ataques adversariais abordados até aqui, tendo em consideração os seguintes aspetos: o ano de desenvolvimento, o tipo de conhecimento, o tipo de especificação, a norma de perturbação utilizada, a força de ataque e o processo para alcançar o resultado.

2.3.3 *Defesas a Ataques Adversariais*

O desenvolvimento e aperfeiçoamento de defesas robustas eficazes exige uma compreensão abrangente de diferentes fatores, bem como avanços contínuos nos métodos de verificação formal de forma a garantir a robustez e a segurança das redes neuronais em aplicações críticas para a segurança, como por exemplo a condução autónoma. Têm sido propostas várias técnicas de defesa contra amostras adversariais para a deteção e classificação de imagens, incluindo as chamadas defesas heurísticas e defesas certificadas [33]. A técnica das defesas heurísticas refere-se a um mecanismo de defesa que possui um bom desempenho na defesa a ataques específicos sem

Ataque	Ano	Tipo de Conhecimento	Tipo de Especificidade	Norma de Perturbação	Força de Ataque	Processo
FGSM [27]	2015	*White-Box*	Direcionado e Não Direcionado	L_∞	Intensidade da perturbação fixada entre $[-\epsilon, +\epsilon]$.	Uma única iteração de forma a se atingir o valor de ruído ϵ definido.
BIM [29]	2017	*White-Box*	Tipicamente Não Direcionado, mas pode ser Direcionado	L_∞	Pode ser instanciado com qualquer região de perturbação para o qual se possa projetar, sendo que a extremidade de área dessa região é determinada tendo em conta o valor de ϵ definido.	Mais do que uma iteração. Utiliza a projeção para se manter dentro da região de perturbação, sendo que a posição inicial da 1ª iteração necessita de ser estabelecida.
PGD [30]	2018	*White-Box*	Tipicamente Não Direcionado, mas pode ser Direcionado	L_2 e L_∞	Pode ser instanciado com qualquer região de perturbação para o qual se possa projetar, sendo que a extremidade de área dessa região é determinada tendo em conta o valor de ϵ definido.	Mais do que uma iteração. Utiliza a projeção para se manter dentro da região de perturbação, mas a posição inicial da 1ª iteração não necessita de ser estabelecida porque a procura de uma perturbação é inicializada num ponto aleatório da região.
Um pixel [32]	2019	*Black-Box*	Tipicamente Não Direcionado, mas pode ser Direcionado	L_∞	Intensidade da perturbação fixa apenas num único pixel.	Utiliza o algoritmo da evolução diferencial, que é um tipo de algoritmo evolucionário, para encontrar o pixel ideal a modificar de modo a maximizar a probabilidade de uma classificação errónea.

Tabela 2.1: Sumário de 4 ataques adversariais.

garantias teóricas exatas. A defesa heurística mais bem sucedida passa pelo treino adversarial, que tenta melhorar a propriedade de robustez de modelos de *Deep Learning* (DL) incorporando exemplos adversariais na fase (ou dados) de treino. Em contrapartida, as defesas certificadas conseguem sempre fornecer certificações até para as técnicas de defesa com precisão mais baixa, sob uma classe bem definida de ataques adversariais. Uma abordagem popular recente de certificação de redes neuronais consiste em formular um politopo adversarial e definir os seus limites superior e inferior utilizando relaxamentos convexos (interpretação abstrata) [23]. Todavia, o desempenho efetivo destas técnicas de defesas certificadas continua a situar-se muito abaixo do desempenho da técnica do treino adversarial.

Consequentemente, em seguida são sumariadas duas técnicas principais de defesa a ataques adversariais que foram desenvolvidas nos últimos anos, dando-se um maior destaque ao treino adversarial porque, como já mencionado anteriormente, é a defesa que até ao momento dispõe de um melhor desempenho em relação à precisão e ao custo computacional [33]:

- Treino Adversarial: O treino adversarial é um método de defesa intuitivo contra amostras adversariais, que tenta melhorar a robustez de um modelo de rede neuronal treinando-a com exemplos adversariais. Formalmente, o treino adversarial é um jogo/problema de otimização *min-max* que pode ser expresso por meio da Equação 2.10 [33]:

$$\min_{\theta} \max_{D(x,x')<\eta} J(\theta, x', y) \tag{2.10}$$

Onde na Equação 2.10, $J(\theta, x', y)$ é a função de perda adversarial, com os valores dos pesos da rede representados por θ, a entrada/exemplo adversarial representada por x' e a saída/*label* verdadeira por y, $D(x,x')$ representa uma métrica de distância entre a entrada benigna x e a entrada adversarial x', e η representa o valor de perturbação adicionada aos dados de entrada x.

O problema de maximização interior ($\max_{D(x,x')<\eta} J(\theta, x', y)$) consiste em encontrar as amostras adversariais mais eficazes, ou seja, consiste em encontrar os exemplos x' que atingem um valor de perda elevada, o que é conseguido através de um ataque adversarial bem concebido, tal como o FGSM [27] e o PGD [30]. Já o problema de minimização externa ($\min_\theta \max_{D(x,x')<\eta} J(\theta, x', y)$) consiste num procedimento de treino padrão de modo a minimizar o valor da perda, isto é, encontra o θ que minimize o maior valor de perda encontrado. A rede resultante será supostamente robusta contra o ataque adversarial utilizado para a geração das amostras adversariais usadas na fase de treino do modelo. Em conclusão, existe um tipo diferente de treino adversarial para cada tipo de ataque aplicado na criação dos exemplos adversariais, *i.e.* um modelo treinado para ser robusto a um ataque FGSM pode não ser robusto a um ataque PGD. Em seguida, é explicitado e ilustrado o treino adversarial FGSM [27], porque é o mais elementar.

Treino Adversarial FGSM

De forma a melhorar a robustez de uma rede neuronal, os investigadores *Goodfellow et al.* propuseram pela primeira vez um princípio de defesa baseado no treino de dados de entrada benignos e amostras adversariais originadas pelo ataque *Fast Gradient Sign Method* [27]. Formalmente, o objetivo do treino adversarial proposto pode ser enunciado através da Equação 2.11 e ilustrado a partir da Figura 2.22 das seguintes formas:

$$\widetilde{J}(\theta, x, y) = cJ(\theta, x, y) + (1 - c)J(\theta, x + \epsilon \cdot sign[\nabla_x J(\theta,x,y)], y) \tag{2.11}$$

Onde na Equação 2.11, $x + \epsilon \cdot sign[\nabla_x J(\theta,x,y)]$ é o exemplo adversarial gerado pelo ataque FGSM, x é a amostra/entrada normal, e a variável c é usada para equilibrar a precisão entre as entradas benignas x e as entradas adversariais x' durante a fase de treino como um hiperparâmetro.

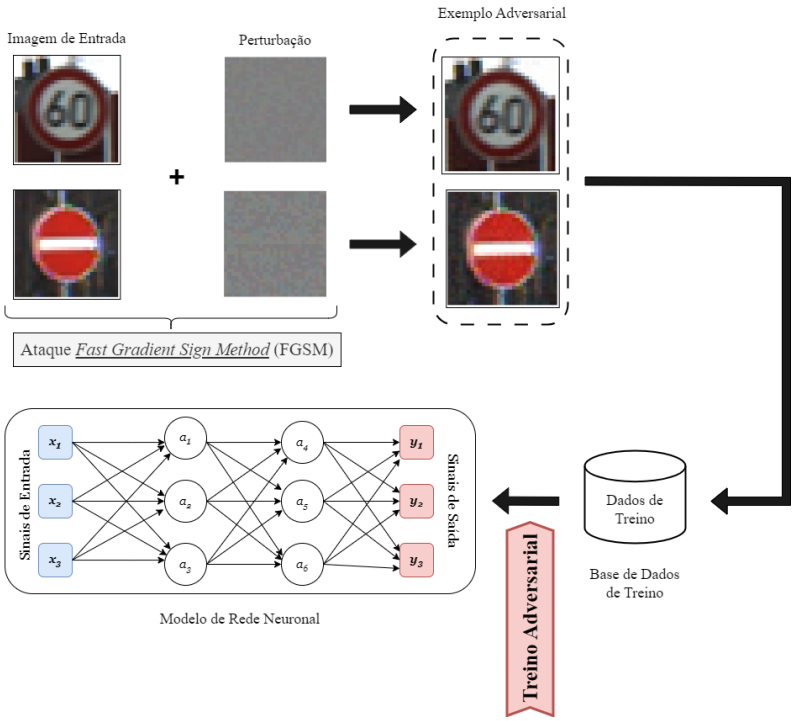

Figura 2.22: O *pipeline* do método de defesa baseado no treino adversarial FGSM.

Pela Figura 2.22, é possível visualizar as diferentes etapas existentes durante o treino adversarial FGSM, isto é, primeiramente são gerados alguns exemplos adversariais a partir da criação de perturbações consoante a intensidade do ataque FGSM, seguidamente os exemplos adversariais produzidos são inseridos/adicionados ao conjunto de dados de treino do modelo, e por fim, ocorre o treino adversarial do modelo de rede neuronal com as amostras benignas da base de dados de treino e também com os exemplos adversariais incorporados.

Os resultados obtidos por *Goodfellow et al.* mostram que uma rede neuronal treinada adversarialmente a ataques FGSM, é capaz de se tornar muito mais robusta porque, especificamente, a taxa de erro de classificação cai de uma forma drástica de 89,4% para 17,9% [27]. No entanto, o modelo de rede neuronal treinado adversarialmente continua a ser vulnerável a ataques adversariais de carácter iterativo ou de otimização, apesar da sua elevada eficácia na defesa de amostras adversariais provocados pelo ataque FGSM;

- Defesas Prováveis: A defesa acima mencionada é uma defesa heurística, o que significa que a sua eficácia só é validada experimentalmente, em vez de ser provada teoricamente. Como as defesas heurísticas não possuem uma garantia teórica relativamente à precisão do erro de classificação, então são suscetíveis de serem ultrapassadas no futuro por um novo ataque que seja desenvolvido. Por conseguiste, este método de defesa provável ainda é novo, e ainda não existem muitos métodos de defesas prováveis *open-source*, mas

> muitos investigadores estão a dedicar cada vez mais esforços para o desenvolvimento de métodos defensivos prováveis contra ataques adversariais, que conseguem sempre assegurar uma garantia teórica de precisão sob uma classe bem definida de ataques [33].

No domínio das defesas, é possível destacar que a comunidade da Inteligência Artificial (AI) e do DL está cada vez mais a se concentrar no desenvolvimento de defesas certificadas, uma vez que a maioria das defesas heurísticas não se conseguem defender contra ataques com um maior nível de complexidade, como por exemplo, os ataques adaptativos *White-Box* [33], e também porque uma defesa certificada é pressuposta de garantir a defesa em condições bem definidas, independentemente do método de ataque aplicado pelo agente adversarial.

No entanto, até agora, a questão da escalabilidade tem sido um problema comum partilhado pela maioria das defesas certificadas. Por exemplo, a análise dos limites dos intervalos de saída de um processo de verificação formal é uma direção recentemente popular utilizada para certificar os modelos de redes neuronais, mas o processo não é dimensionável para redes neuronais profundas e grandes conjuntos de dados [23, 33]. Assim sendo, é evidente que, em comparação com os ataques adversariais, o aperfeiçoamento e o desenvolvimento das defesas enfrentam muito mais obstáculos, sendo que isso deve-se, principalmente, ao facto de um ataque adversarial só poder visar um tipo ou categoria de defesas, mas as defesas necessitam de ser certificadas, ou seja, necessitam de ser eficazes para um qualquer tipo ou método de ataque possível em determinadas situações.

Esta página foi propositadamente deixada em branco.

3

ESTADO DA ARTE

O presente capítulo, apresenta o estado da arte no que respeita a implementação de técnicas e métodos de ML e DL no enquadramento da condução autónoma. São identificados alguns dos setores e possíveis aplicações onde a utilização de algoritmos de ML e DL são utilizados para o fenómeno da condução autónoma, dando-se mais importância ao campo da perceção e modelação do meio ambiente do veículo, sendo também enunciados alguns trabalhos desenvolvidos ao longo do tempo para esse tópico. Posteriormente, são apresentados alguns trabalhos pertinentes na área específica do reconhecimento e classificação de sinais de trânsito e as abordagens tipicamente utilizadas para o objetivo em questão. Finalmente, é realizado um breve resumo (visão geral) sobre a competição internacional de verificação formal de redes neuronais (do inglês, *Verification of Neural Networks Competition* (VNN-COMP)), mais especificamente, a sua 4ª edição que se refere ao ano de 2023, porque esta competição é um dos motivos e fundamentos a partir do qual surge a ideia deste trabalho dada a crescente implantação de sistemas de ML e DL em domínios críticos como a condução autónoma, onde são essenciais as garantias formais de funcionamento.

Condução Autónoma

Na última década, registou-se um crescimento progressivo da tecnologia dos veículos autónomos, apoiado principalmente pelos avanços na área do *Deep Learning* (DL), *Machine Learning* (ML) e da Inteligência Artificial (AI). Por essa razão, os veículos autónomos já migraram das fases de desenvolvimento e teste em laboratório para a fase de condução real em estradas públicas, sendo que a sua implementação proporciona uma diminuição dos acidentes rodoviários e dos congestionamentos de trânsito, bem como uma melhoria significativa na mobilidade em cidades sobrelotadas [34].

O título **"Condução Autónoma"** pode parecer um conceito evidente, mas segundo a Sociedade dos Engenheiros Automóveis (do inglês, *Society of Automotive Engineers* (SAE)), é possível definir seis níveis diferentes de automação. Como se pode observar pela Tabela 3.1, a norma SAE J3016 [35] introduz uma escala entre 0 e 5 para classificação do tipo de automatização dos veículos. Enquanto os níveis SAE mais baixos apresentam uma assistência mais básica ao condutor, os níveis SAE mais elevados evoluem para tipos de veículos que não requerem qualquer tipo de interação humana, de tal forma que os automóveis da categoria de nível SAE 5 não

necessitam de qualquer intervenção ou monitorização do condutor e, normalmente, nem sequer possuem volante ou pedais como os automóveis normais.

Os veículos de condução autónoma são sistemas autónomos de tomada de decisões que processam fluxos de informações provenientes de diferentes fontes incorporadas a bordo, tais como câmaras, RADAR, LiDAR, sensores ultrassónicos, unidades *Global Positioning System* (GPS) e/ou sensores de inércia, sendo que estas observações são utilizadas pelo "computador" do automóvel de modo a tomar decisões de condução. O diagrama de blocos básicos de um veículo autónomo é apresentado na Figura 3.1, e este é representado como um *pipeline* modular de perceção-planeamento-ação [34,36]:

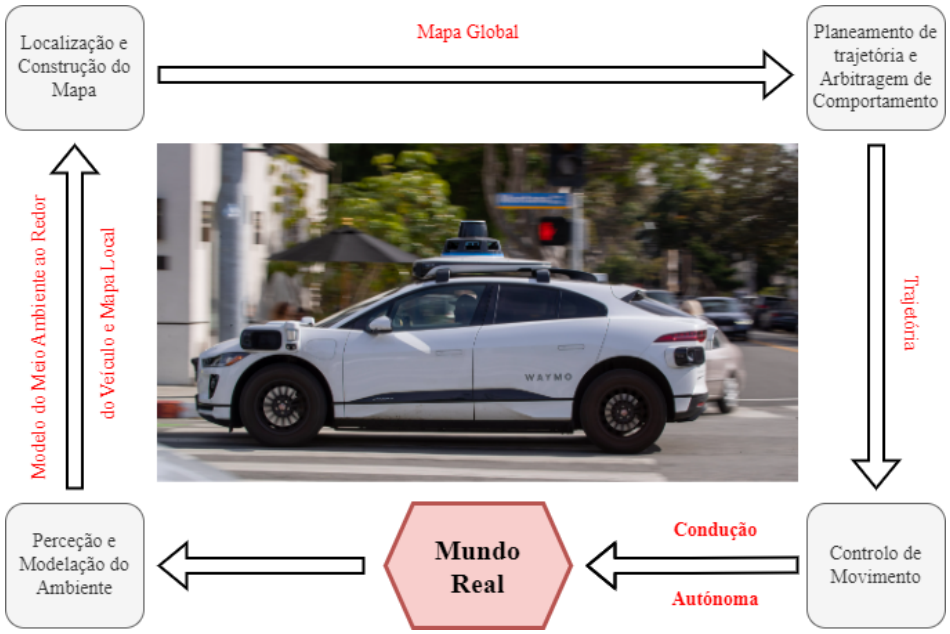

Figura 3.1: Diagrama de blocos representativo de um veículo autónomo com as 4 tecnologias fundamentais. Adaptada de [36]

Como se verifica na Figura 3.1 [36], o diagrama é decomposto hierarquicamente em quatro componentes (ou tecnologias) que podem ser concebidas utilizando abordagens de DL e AI, ou métodos clássicos. Estes componentes são:

- Perceção e modelação do ambiente ao redor do veículo;
- Localização e construção do mapa;
- Planeamento de trajetória e arbitragem de comportamento;
- Controladores de movimento.

Desta forma, dada uma rota planeada, a primeira tarefa de um veículo autónomo é compreender e localizar-se no meio circundante, sendo que com base nesta representação, é planeado um caminho contínuo e as ações futuras do veículo são determinadas pelo seu sistema de arbitragem de comportamento ou tomada de

Nível de Automatização e Definição	Descrição
Nível 0 Sem Automatização de Condução	Algumas funcionalidades e sistemas de apoio à condução como o controlo de estabilidade (do inglês, *Eletronic Stability Control* (ESC)), o assistente de travagem de emergência (do inglês, *Brake Assist System* (BAS)), o *Cruise Control* (CC), o sistema de travagem anti-bloqueio (do inglês, *Anti-lock Braking System* (ABS)) e o controlo de tração (do inglês, *Anti Slip Regulation* (ASR)) estão inseridos neste nível de automação. Trata-se, portanto, de sistemas totalmente não automatizados, dando o controlo completo ao condutor
Nível 1 Assistência ao Condutor	O condutor mantém a responsabilidade de prestar atenção a outros veículos e estar sempre atento quando estes travarem. Um exemplo é o sistema de *Cruise Control* adaptativo, que ajusta a velocidade mediante a distância ao veículo em frente, permitindo ao condutor levantar temporariamente o pé dos pedais, sendo que o condutor tem sempre a decisão absoluta do controlo do veículo
Nível 2 Automatização Parcial de Condução	O veículo consegue combinar as dinâmicas longitudinais com as laterais, de forma a ajustar a sua posição na faixa de rodagem com o controlo da velocidade. De notar que, apesar do veículo conseguir "conduzir" sozinho, o controlo é da inteira responsabilidade do condutor, que deve monitorizar o veículo durante todo o momento
Nível 3 Automatização Condicional de Condução	O automóvel já consegue assumir o controlo em situações mais específicas, realizando algumas tarefas sem a intervenção do condutor, tal como por exemplo, o estacionamento automático ou a capacidade de conduzir em vias com muito trânsito. Mas, mais uma vez, o condutor continua a ser necessário para intervir em situações que sejam mandatárias
Nível 4 Automatização Elevada de Condução	O veículo é capaz de assumir o controlo de todas as principais componentes da condução. Este consegue conduzir de forma autónoma, mas em percursos pré-definidos, como por exemplo, os táxis inteligentes da Waymo que já estão a ser testados nos Estados Unidos, sem recurso a um condutor
Nível 5 Automatização Total de Condução	Apesar de ainda não ter sido alcançado, espera-se que este nível máximo de automação acrescente o fator do veículo conseguir conduzir de forma totalmente autónoma, sem qualquer rota predefinida ou intervenção humana quando este se desvia do percurso original. O veículo deve conseguir conduzir em qualquer circunstância ou situação de tráfego

Tabela 3.1: Diferentes níveis SAE de automatização da condução autónoma. Adaptada de [35]

decisão. Finalmente, o sistema de controlo de movimento corrige automaticamente os possíveis erros gerados na execução do movimento programado [34].

O primeiro e um dos principais requisitos para os veículos inteligentes (ou autónomos) é o facto de terem de possuir a capacidade de perceber e compreender o meio que os rodeia em tempo real. Tal como mencionado previamente, os dados são normalmente recolhidos por vários sensores, tais como a câmara, o RADAR, o LiDAR e outros tipos, sendo que após o pré-processamento desses dados, são extraídas várias características de objetos existentes no meio envolvente do veículo, como estradas, as linhas da faixa de rodagem, sinais de trânsito, peões e outros veículos. Desta maneira, tanto os objetos estáticos como os em movimento do ambiente ao redor do veículo podem ser detetados e localizados, sendo possível analisar o comportamento do veículo e proceder à interpretação da situação [36].

Uma representação em forma de diagrama da estrutura da tecnologia de perceção e modelação do ambiente de um veículo autónomo é apresentada na Figura 3.2 [36]:

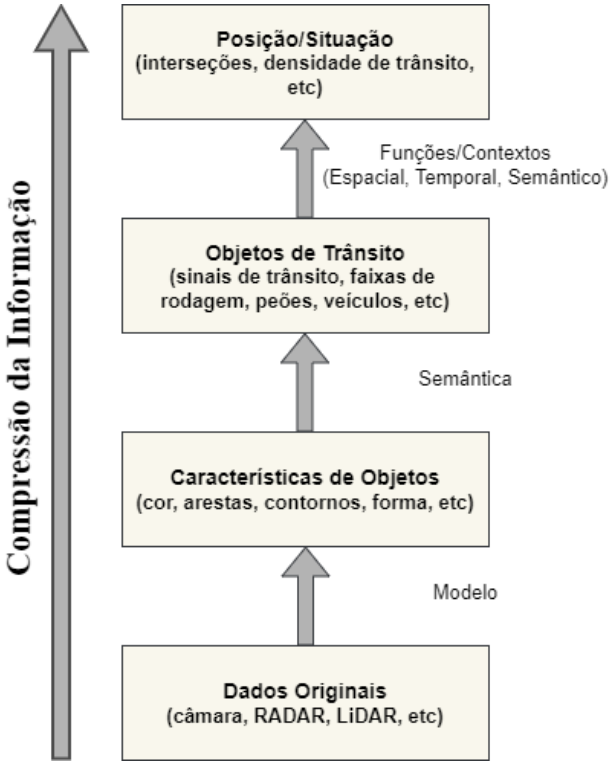

Figura 3.2: Esquema da tarefa de perceção e modelação do meio ambiente de um veículo autónomo. Adaptada de [36]

A partir da Figura 3.2, é possível reconhecer que as principais funções da componente da perceção e modelação do ambiente para veículos inteligentes baseiam-se na deteção e classificação de objetos de trânsito, isto é, faixas de rodagem, peões, veículos, e sinais de trânsito, na análise do seu comportamento, bem como na interpretação do cenário.

Consequentemente, em seguida, é apresentado um resumo de alguns trabalhos realizados por diversos autores das abordagens e das técnicas mais utilizadas na perceção do ambiente dos veículos autónomos, mais precisamente na verificação formal de métodos para a deteção de estradas, das linhas da faixa de rodagem, de pedestres e de veículos (Capítulo 3.1), e do reconhecimento e classificação de sinais de trânsito (Capítulo 3.2).

3.1 DETEÇÃO DE ESTRADAS, LINHAS DE FAIXAS DE RODAGEM, VEÍCULOS E PEÕES

No âmbito da visão computacional aplicada à condução autónoma, a deteção de estradas, linhas de faixa de rodagem, veículos e peões constitui um desafio essencial para garantir a segurança e eficiência dos sistemas de transporte. Contudo, uma classificação precisa de sinais de trânsito representa um componente crítico com impacto direto nas decisões de navegação e na conformidade das ações realizadas pelo veículo autónomo.

Desta forma, foi realizado um estudo e uma revisão bibliográfica mais extensa com um foco particular para a tarefa de reconhecimento e classificação de sinais de trânsito (Capítulo 3.2) relativamente à tarefa de deteção de estradas, linhas de faixa de rodagem, veículos e peões deste subcapítulo por várias razões:

- Papel central dos sinais de trânsito, porque encapsulam regras e informações que condicionam a dinâmica de tráfego, onde uma interpretação incorreta ou a ausência de uma classificação adequada de um sinal de trânsito pode levar a infrações ou acidentes graves, colocando vidas em risco;
- Complexidade e variabilidade, porque diferente de outros elementos nas vias públicas, os sinais de trânsito apresentam uma enorme variabilidade em termos de design, idioma, iluminação e ângulos de visão, sendo que essa complexidade exige métodos formais rigorosos de modo a garantir a robustez e a confiabilidade do sistema de classificação;
- Complementaridade com outras tarefas, porque a deteção de estradas, linhas de faixas de rodagem, veículos e peões forma a base da perceção ambiental ao redor do veículo, mas a correta interpretação dos sinais de trânsito fornece o contexto necessário para priorizar e decidir ações, como paragens, mudanças de direção ou ajustes de velocidade.

3.1.1 *Deteção de Estradas e Linhas de Faixas de Rodagem*

A deteção de estradas e linhas de faixa de rodagem é um campo de investigação ativo no domínio da condução autónoma. Atualmente, existem quatro sistemas de deteção de estradas e das linhas da faixa de rodagem aplicadas nos veículos, sendo eles o sistema de aviso de saída da faixa de rodagem, o *Cruise Control* Adaptativo (do inglês, *Adaptive Cruise Control* (ACC)), o sistema de manutenção ou centragem da faixa de rodagem e o sistema de assistência à mudança de faixa. Seguindo a estrutura da tecnologia de perceção e modelação do ambiente (da Figura 3.2), o sistema de deteção de estradas e linhas de faixa de rodagem é normalmente constituído por três componentes, isto é, o pré-processamento, a extração de características e o ajuste do modelo [36].

A dificuldade de um sistema de deteção das linhas da faixa de rodagem e estradas é a diversidade de condições, como a diversidade que existe no aspeto das linhas e das estradas, a nitidez da imagem e a fraca visibilidade. Por conseguinte, de modo a melhorar o desempenho da deteção foram propostos vários algoritmos de acordo com diferentes pressupostos sobre uma possível estrada estrutural, sendo que os métodos referenciados em seguida, aplicam no mínimo um ou mais destes pressupostos [36]:

1. A textura da estrada/faixa de rodagem é consistente;
2. A largura da estrada/faixa de rodagem é localmente constante;
3. A marcação da estrada (linhas) segue regras restritas de aparência ou colocação;
4. A estrada é um plano ou segue um modelo rigoroso de alteração da elevação.

Um dos principais problemas da maioria dos algoritmos de verificação formal de extração de características existentes na deteção de estradas e faixas de rodagem é que a informação global sobre a forma da geometria da faixa de rodagem não é incluída. Em situações complexas em que a estrada está cheia de sombras ou outros objetos, o conteúdo da imagem original em vários locais partilha propriedades semelhantes com as linhas/limites da faixa de rodagem. Caso o algoritmo de extração de características se basear apenas numa área local da imagem, então serão detetados pontos de características indesejadas e estas características serão depois incluídas durante a fase de estimativa e localização dos parâmetros (estrada e linhas), causando imprecisões e erros. Assim, estes pontos de características não desejadas devem ser excluídos antes do processo de estimação dos parâmetros [37].

A partir do problema acima referido, *Wang et al.* [37] propôs uma nova abordagem de extração de características da faixa de rodagem e estradas num trabalho desenvolvido pelos autores. A abordagem de extração de características proposta baseia-se

Estudo de Investigação e Ano	Características	Resultados quantitativos indicados no documento original	Comentário
Wang et al., 2012 [37]	Extração de características através da informação global sobre a forma da estrada/faixa de rodagem e ajuste do modelo através de parábolas paralelas no plano do solo	Base de dados: várias sequências de vídeo. Avaliador de desempenho: O valor de ER, que significa erro de estimativa dos parâmetros. Resultados: - cenário 1 (Figura 3.3(a)) ER: 0.11 (parâmetro a), 0.27 (parâmetro b_1), 0.38 (parâmetro b_2), e 0.71 (parâmetro c); - cenário 2 (3.3(d)) ER: 0.36 (parâmetro a), 0.99 (parâmetro b_1), 0.27 (parâmetro b_2), e 0.80 (parâmetro c); - cenário 3 (3.3(g)) ER: 1.06 (parâmetro a), 0.41 (parâmetro b_1), 0.19 (parâmetro b_2), e 1.01 (parâmetro c).	O método tem como base o facto de que, ao se realizar *zoom* no ponto de desvanecimento das linhas da faixa de rodagem, as marcações só se irão deslocar nas mesmas linhas retas em que se encontram. O parâmetro a controla a curvatura das linhas. Os parâmetros b_1 e b_2 correspondem às posições horizontais das linhas da esquerda e da direita das curvas. O parâmetro c representa a posição do ponto de desvanecimento
Chen et al., 2021 [38]	Três arquiteturas baseadas em redes neuronais convolucionais e seis extratores de características treinados na base de dados KITTI para deteção de veículos e peões	Base de dados: conjuntos de imagens da KITTI. Avaliador de desempenho: precisão média, velocidade, consumo/alocação de memória e número de parâmetros. Resultados: - *Faster R-CNN ResNet50* tem a melhor precisão média (58%) para a deteção de veículos e peões, com uma velocidade de 8.6 FPS; - *Faster R-CNN Inception_V2* possui o melhor desempenho na deteção de veículos e na deteção de peões (74.5% e 47.3%, respetivamente); - *ResNet101* consome a maior quantidade de memória (9907 MB) e tem o maior número de parâmetros (64.42 milhões); - *Inception_ResNet_V2* é o modelo mais lento (3.05 FPS); - *SSD MobileNet_V2* é o modelo mais rápido (70 FPS); - *SSD MobileNet_V1* é o modelo mais leve em termos de utilização de memória (875 MB).	Como o trabalho é um estudo, então não é desenvolvido nenhum algoritmo novo, mas através deste estudo é possível entender quais são as arquiteturas que existem até ao momento para deteção de objetos, e também qual a melhor para a deteção de veículos e peões. Como o modelo *Faster R-CNN ResNet50* atinge a precisão média mais elevada, então tem o melhor equilíbrio entre velocidade de processamento e precisão. Como o modelo *SSD MobileNet_V2* é o mais rápido e o modelo *SSD MobileNet_V1* é o mais leve em termos de memória, então são ambos compatíveis para aplicações em dispositivos móveis e embebidos
Demarchi et al., 2022 [39]	Três arquiteturas de redes neuronais, uma saída: a aceleração $a[m/s^2]$ sugerida ao veículo *ego* e seis entradas: a velocidade $v_p[m/s]$ do veículo *ego*, a velocidade $S_r[m/s]$ do veículo *exo* relativamente ao veículo *ego*, a distância atual $D[m]$ entre o veículo *ego* e o veículo *exo*, o tempo mínimo de avanço $TH[s]$ que é o intervalo de tempo mínimo entre o veículo *exo* e o veículo *ego*, a distância mínima de segurança $D_s[m]$ que é igual a $TH \cdot v_p$ e a margem $D_o[m]$ de segurança a ser adicionada à distância mínima de segurança D_s	Base de dados: não especificado no documento. Avaliador de desempenho: verificabilidade das três propriedades definidas *OutBounds*, *Near0* e *Far0*. Resultados: - Todas as propriedades conseguem ser verificadas em todas as redes num intervalo de tempo razoável, é necessário menos de 1 minuto de tempo de CPU, independentemente da arquitetura da rede para a verificação estar completa; - Para o caso de TH de 1.5 segundos e D_o de 5 metros, a propriedade *OutBounds* é sempre verdadeira para as redes *Net0*, *Net1* e *Net2*, e as propriedades *Near0* e *Far0* são sempre falsas para as redes *Net0*, *Net1* e *Net2*; - Para o caso de TH de 1.5 segundos e D_o de 25 metros, a propriedade *OutBounds* continua a ser sempre verdadeira para as redes *Net0*, *Net1* e *Net2*, a propriedade *Near0* continua a ser sempre falsa para as redes *Net0* e *Net1*, mas passa a ser maioritariamente falsa (existem algumas iterações de teste onde é verdadeira) para a rede *Net2*, e a propriedade *Far0* continua a ser sempre falsa para as redes *Net1* e *Net2*, mas passa a ser maioritariamente verdadeira para a rede *Net0*.	O algoritmo/modelo de *Cruise Control* Adaptativo desenvolvido, pode-se dizer que é um dos exemplos concretos de verificação formal que mais se aproxima da aplicação industrial no setor automóvel. São mostrados resultados experimentais convincentes de que é possível aprender uma função tão complexa de controlo de ACC e verificar algumas propriedades relevantes. Denota-se que este trabalho é um exemplo introdutório porque só foram realizados dois testes experimentais

Tabela 3.2: Métodos de deteção de estradas, linhas de faixa de rodagem, peões e veículos.

no facto de que, ao se realizar uma aproximação (*zoom*) no ponto de desvanecimento da faixa de rodagem na imagem de teste/original, as linhas da faixa de rodagem apenas se irão mover nas mesmas linhas retas em que se encontram, e dessa forma, os objetos/características que não sejam as linhas da faixa de rodagem na imagem não partilham dessa propriedade e, por isso, podem ser ignorados durante a deteção dos parâmetros do modelo. Os resultados obtidos podem ser caracterizados como quantitativos e qualitativos. Enquanto os resultados quantitativos estão enunciados na Tabela 3.2, que resume os vários métodos de verificação formal estudados para a deteção de estradas, linhas de faixa de rodagem, veículos e peões, os resultados qualitativos são enunciados na Figura 3.3. Como se pode verificar na Figura 3.3, as imagens de entrada utilizadas foram escolhidas entre as cenas mais difíceis de várias sequências de vídeo já que, os cenários das imagens (a) e (d) são fortemente afetados por sombras, e o cenário da imagem (g) é um cenário de uma faixa de rodagem que diverge. Já os mapas de gradiente correspondentes às imagens (a), (d) e (g) são apresentados, respetivamente, nas Figuras 3.3 (b), (e) e (h), sendo que todos esses mapas de gradiente contêm um grande número de pontos de características indesejadas. As Figuras 3.3 (c), (f) e (i) apresentam o mapa de características obtido utilizando o algoritmo proposto pelo trabalho na literatura, e a maioria das características indesejadas são removidas. Em comparação com os mapas de gradiente, os mapas de características possuem um menor nível de ruído e as características da faixa de rodagem (linhas) são muito melhor preservadas [37].

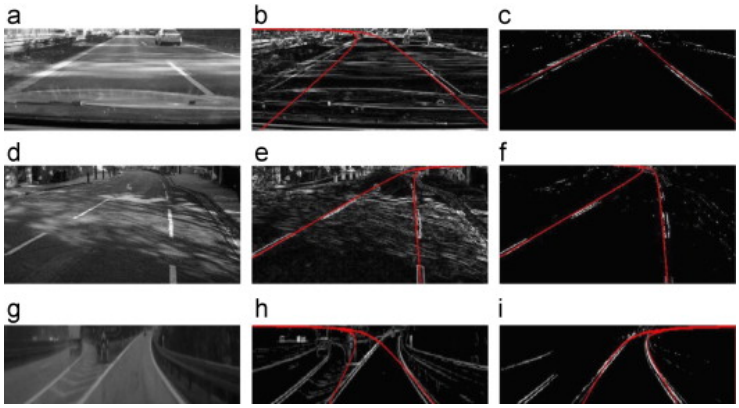

Figura 3.3: Resultados da estimativa dos parâmetros do modelo da faixa de rodagem. Retirada de [37]

3.1.2 *Deteção de Veículos e Peões*

As tecnologias deste tipo de deteção são normalmente utilizadas em aplicações de condução autónoma para fins de segurança pública. Por exemplo, a deteção de veículos é a premissa do planeamento e controlo de veículos autónomos, já que só coordenando a relação posicional com os veículos circundantes, é que é possível

garantir a segurança do veículo autónomo. No mesmo sentido, a deteção de peões é utilizada de modo a proteger os próprios peões, evitando assim prováveis colisões do veículo autónomo contra eles. A investigação sobre métodos de verificação formal de deteção de veículos e peões enfrenta dois problemas, o primeiro relativo aos ambientes exteriores complexos, como por exemplo a iluminação, as alterações de fundo e as oclusões que existem no meio ambiente, e o segundo devido à falta de consenso na seleção de um algoritmo de deteção que contenha um equilíbrio adequado entre a precisão da deteção, a velocidade e o consumo de memória [36,38].

Desta forma, com a resolução dos dois problemas mencionados acima em mente, os autores *Chen et al.* [38] desenvolveram um estudo que realiza a comparação de diferentes arquiteturas com diferentes extratores de características típicos para a deteção de objetos, mais rigorosamente, para a deteção de veículos e pedestres. Neste trabalho, são implementadas e avaliadas particularmente três arquiteturas baseadas em redes neuronais convolucionais (CNN) chamadas de *Faster Region-based Convolutional Neural Network* (R-CNN), *Region-based Fully Convolutional Network* (R-FCN) e *Single Shot Detector* (SSD). Também são utilizados seis modelos de extratores de características, chamados de *ResNet50*, *ResNet101*, *MobileNet_V1*, *MobileNet_V2*, *Inception_V2* e *Inception_ResNet_V2*, obtendo-se ao todo oito algoritmos/modelos. Todos os modelos são pré-treinados na base de dados COCO, sendo que de modo a possuírem um melhor desempenho na deteção de veículos e peões, os modelos também são posteriormente aperfeiçoados com a base de dados KITTI [40], que contém imagens específicas para o problema de deteção de objetos no domínio da condução autónoma. Os resultados obtidos, tal como no trabalho anterior de deteção de estradas e linhas de faixa de rodagem, são quantitativos e qualitativos. Os resultados quantitativos estão resumidos na Tabela 3.2, sendo que os oito modelos de deteção de veículos e peões são avaliados através de métricas padrão, tais como a precisão média, o tempo de execução, a atribuição de memória e o número de parâmetros de rede. Já os resultados qualitativos estão demonstrados na Figura 3.4. Na Figura 3.4, é apresentada a comparação qualitativa para a deteção de veículos e pedestres em imagens da base de dados KITTI, utilizando os modelos previamente indicados, sendo que são representados os resultados da deteção de três cenários comuns, isto é, cenários que contêm apenas veículos (imagens (a) e (b)), veículos e peões (imagens (c) e (d)) e apenas peões (imagens (e) e (f)). No geral, é possível deduzir que todos os modelos têm um bom desempenho na deteção de veículos nas imagens (a) e (b), e nas imagens (c) e (d). A única exceção é o modelo *SSD MobileNet* que não conseguiu detetar a pessoa na imagem (c). No entanto, na imagem (d), é possível verificar que os modelos *Faster R-CNN ResNet101*, *R-FCN ResNet101*, *SSD MobileNet_V1* e *SSD Inception_V2* ignoram a pessoa no lado direito da imagem, mas também, como o cenário apresentado na imagem (d) é o mais complexo, então é pouco provável que os modelos consigam detetar com toda

a precisão objetos parcialmente obscurecidos. Finalmente, nas imagens (e) e (f), todos os modelos apresentam dificuldades em detetar todos os peões, especialmente o modelo *SSD MobileNet_V1*.

Figura 3.4: Comparação qualitativa dos diferentes modelos na base de dados KITTI. Retirada de [38]

Um dos exemplos de deteção de veículos mais disseminados no setor automóvel, é o *Cruise Control* Adaptativo (ACC), sendo que esta tecnologia encontra-se classificada como uma função de condução autónoma de nível SAE 1 (Tabela 3.1) que controla a aceleração do veículo autónomo, ou também se podendo chamar de veículo *ego* (veículo no qual o ACC está instalado), ao longo do eixo longitudinal. Um sistema de ACC tem dois objetivos concorrentes, que são manter o veículo *ego* à velocidade definida pelo utilizador (modo de velocidade de cruzeiro) e manter uma distância de segurança relativamente a um possível veículo em frente, sendo que esse veículo também se pode chamar de veículo *exo* (modo de seguimento) [39].

O trabalho desenvolvido por *Demarchi et al.* [39] pretende desenvolver um método de verificação e aprendizagem de uma rede neuronal que replica a função de controlo de um sistema de ACC semelhante aos utilizados nos automóveis atuais. O objetivo desse trabalho foi manter o veículo autónomo a uma velocidade definida pelo utilizador e, eventualmente, adaptar a própria velocidade tendo em conta outros possíveis veículos que possam circular em frente do veículo *ego*. Para isso, o modelo

de ACC considerado contém uma saída e seis entradas, onde ambas as entradas e saída do modelo estão definidas na coluna das "Características" da Tabela 3.2. No que se refere à parte experimental do algoritmo desenvolvido no artigo, foram testadas três arquiteturas de redes neuronais compostas por camadas afins (*i.e. affine layers*) e ReLU, que apresentam uma complexidade crescente, tanto em termos do número de camadas como da quantidade de neurónios por camada, isto é, a rede neuronal *Net0* possui duas camadas afins de 20 e 10 neurónios respetivamente, sendo cada uma seguida de uma camada ReLU, a rede neuronal *Net1* possui duas camadas afins de 50 e 40 neurónios respetivamente, sendo cada uma seguida de uma camada ReLU, e a rede neuronal *Net2* possui quatro camadas afins de 20, 20, 20 e 10 neurónios respetivamente, sendo cada uma seguida de uma camada ReLU. Para o caso de estudo de verificação do modelo de ACC desenvolvido com diferentes redes neuronais, também foram consideradas e definidas três propriedades. A primeira propriedade definida é designada por *OutBounds*, e verifica simplesmente se a aceleração de saída não excede os limites da função/modelo de ACC utilizada, ou seja, se por exemplo a aceleração de saída satisfaz a pós-condição $-3 \leqslant a \leqslant 1$. A segunda propriedade é identificada por *Near0* e tem como objetivo garantir que a função ACC não produz acelerações positivas quando o veículo *exo* se encontra demasiado próximo do veículo *ego*, isto é, se por exemplo a aceleração de saída satisfaz a pós-condição $-3 \leqslant a \leqslant 0$. Finalmente, a última propriedade definida é designada por *Far0*, e é simétrica em relação à propriedade *Near0*, ou seja, pretende verificar que quando o veículo *exo* se encontra demasiado longe do veículo *ego*, então o modelo de ACC não sugere acelerações negativas, isto é, se por exemplo a aceleração de saída satisfaz a pós-condição $0 \leqslant a \leqslant 1$. Os resultados obtidos, diferenciando com os dois trabalhos de deteção de estradas, linhas de faixa de rodagem, veículos e peões explicados anteriormente, são só quantitativos, e encontram-se resumidos na Tabela 3.2.

3.2 RECONHECIMENTO E CLASSIFICAÇÃO DE SINAIS DE TRÂNSITO

O reconhecimento de sinais de trânsito (do inglês, *Traffic Sign Recognition* (TSR)) é um problema de classificação multi-classe de bases de dados com frequências de classe desequilibradas. O TSR divide-se, atualmente, em duas tarefas, isto é, a deteção de sinais de trânsito (do inglês, *Traffic Sign Detection* (TSD)) que visa a localização de sinais de trânsito numa imagem de entrada, e a classificação de sinais de trânsito (do inglês, *Traffic Sign Classification* (TSC)) que tem o objetivo de prever os *labels* de cada classe diferente de sinais de trânsito para as deteções [41].

O correto reconhecimento de sinais de trânsito, que é realizado predominantemente com redes neuronais profundas, principalmente as CNN, é vital para as tarefas de

Base de dados (ano de publicação)	País	TSC	TSD	Imagens Anotadas	Classes	Resolução
GTSRB (2011) [5]	Alemanha	✓	✗	50k+	43	15x15 até 250x250
BelgiumTS (2011) [42]	Bélgica	✓	✗	9k	62	11x10 até 562x438
LISA (2012) [43]	Estados Unidos da América	✓	✓	6k	47	640x480 até 1024x522
TT100K (2016) [44]	China	✓	✓	100k	221	2048x2048

Tabela 3.3: Visão geral de 4 bases de dados de reconhecimento de sinais de trânsito. Adaptada de [41]

perceção e modelação em veículos autónomos. Ao contrário de outras tarefas de perceção e modelação, o TSR só pode ser efetuado utilizando imagens de câmaras, o que significa que os erros ou falhas não podem ser compensados por outros sensores como o LiDAR e o RADAR. Os ataques adversariais aos modelos de ML e DL representam, assim, uma ameaça especial para a robustez e precisão da tarefa de reconhecimento de sinais de trânsito [41].

Assim, neste subcapítulo são referenciados alguns trabalhos de ataques adversariais, classificadores, bases de dados e outras técnicas relevantes para analisar e testar a propriedade de robustez ou precisão dos modelos de classificação de sinais de trânsito desenvolvidos.

3.2.1 *Bases de Dados*

Os sinais de trânsito de diferentes países e continentes apresentam grandes variações em termos de forma e contexto, sendo que por essa razão, muitos trabalhos optam por utilizar bases de dados diferentes, pois os autores geralmente optam por utilizar a base de dados de sinais de trânsito mais similar à do país onde se encontram a realizar o estudo. Tendo em conta este facto, na Tabela 3.3 encontram-se apresentadas e resumidas as bases de dados mais utilizadas e conhecidas para a tarefa de classificação de sinais de trânsito [41].

Devido a ser uma das base de dados mais antiga e com um dos maiores números de imagens anotadas, a GTSRB continua a ser a mais utilizada para a avaliação dos algoritmos de TSC, apesar de as bases de dados *Belgium Traffic Signs* (BelgiumTS) e *LISA* também serem muito utilizadas. Já as abordagens mais recentes são, frequentemente, avaliadas na base de dados *TsinghuaTencent 100K* (TT100K), uma vez que comparativamente à base de dados GTSRB, possui um maior número de imagens anotadas e um maior número de classes de sinais de trânsito. Adicionalmente, é mais flexível, pois ao usar esta base de dados as tarefas de TSC e TSD podem ser combinadas.

3.2.2 *Ataques Adversariais*

Nesta secção, são apresentados e analisados alguns ataques adversariais existentes para a classificação de sinais de trânsito, principalmente os mais realistas, *e.g.* gotas de chuva, nevoeiro e/ou sombras. A Tabela 3.4 apresenta uma visão geral mais objetiva dos ataques referenciados em seguida, e a Figura 3.5 demonstra alguns exemplos de imagens de sinais de trânsito manipulados com os ataques citados.

Um dos primeiros trabalhos sobre ataques adversariais aplicados no domínio do reconhecimento de sinais de trânsito surgiu em 2017 [45]. O trabalho desenvolvido pelos autores *Eykholt et al.* [45] centrou-se em ataques existentes no mundo real contra modelos de TSC, e para isso, foi proposta e desenvolvida a abordagem de ataque *White-Box* chamada de perturbações físicas robustas (do inglês, *Robust Physical Perturbations* (RP2)), em que uma perturbação é obtida a partir de uma distribuição de perturbações fisicamente possíveis com o objetivo de maximizar o erro de classificação. O ataque, como se é normalmente observado em sinais de trânsito reais, foi concebido de modo a se assemelhar a um *graffiti*, que é uma arte urbana realizada por seres humanos. Para tal, foi aplicada à imagem de entrada uma máscara como forma de melhoramento da eficácia do ataque, porque desta maneira, é criada uma perturbação visível, mas discreta, que perturba apenas o objeto (sinal de trânsito) e não o ambiente ao seu redor. Um exemplo de uma manipulação resultante do ataque RP2 que consiste num conjunto de autocolantes a preto e branco aplicados digitalmente a um sinal de STOP está ilustrado na Figura 3.5a. De forma a ser possível verificar a força do ataque implementado e a taxa de erro de classificação, os autores utilizaram dois modelos distintos de redes neuronais convolucionais, isto é, a *LISA*-CNN que consiste em três camadas convolucionais seguidas de uma camada totalmente conectada treinada através da base de dados *LISA*, e a GTSRB-CNN que consiste em três camadas convolucionais seguidas de duas camadas totalmente conectadas treinada na base de dados GTSRB. Além disso, os autores *Eykholt et al.* também propuseram uma nova metodologia de avaliação dos ataques aplicados no mundo real em duas fases, ou seja, experiências estacionárias em laboratório com a utilização das imagens de base de dados e a avaliação/testes no "terreno" utilizando cenários de condução gravados em vídeo.

Mais recentemente, os trabalhos desenvolvidos de ataques adversariais aplicados na tarefa do TSR estão mais empenhados em recriar ataques de fenómenos naturais, tais como sombras [46, 47], manchas [46] e gotas de chuva [48] implementadas nos sinais de trânsito.

Um exemplo disso é o ataque proposto pelos autores *Yang et al.* [46], denominado de ataque direcionado de atenção (do inglês, *Targeted Attention Attack* (TAA)) que aplica um mecanismo de atenção de modo a se determinar quais são as áreas (pixeis)

Autores (ano de publicação)	Base(s) de dados	Modelo(s) de rede neuronal	Método e Aspeto de Ataque	TSC	TSD	White-Box	Black-Box	Mundo Real
Eykholt et al. (2017) [45]	LISA, GTSRB	LISA-CNN, GTSRB-CNN	RP2 em forma de autocolantes a preto e branco	✓	✗	✓	✗	✓
Yang et al. (2021) [46]	LISA, GTSRB	CNN de 3 camadas	TAA em forma de manchas e/ou sombras	✓	✗	✓	✗	✓
Zhong et al. (2022) [47]	LISA, GTSRB	LISA-CNN, GTSRB-CNN	EOT em forma de sombras	✓	✗	✗	✓	✓
Liu et al. (2023) [48]	GTSRB, TT100K	GAN	GAN para gerar gotas de chuva (AdvRD)	✓	✓	✗	✓	✓

Tabela 3.4: Visão geral de 4 ataques adversariais aos modelos de reconhecimento de sinais de trânsito. Adaptada de [41]

do sinal de trânsito em teste que são especialmente mais favoráveis à colocação de uma sombra ou mancha. O objetivo do ataque é, assim, encontrar os pixeis mais sensíveis de modo a otimizar a perturbação aplicada, sendo que esta perturbação é facilmente ignorada pelo condutor real, mas conduz a uma elevada taxa de má classificação de sinais de trânsito no modelo utilizado. O modelo de rede neuronal desenvolvido pelos autores é uma rede neuronal de três camadas convolucionais, que é treinada a partir das bases de dados GTSRB e *LISA*, e os resultados experimentais deste trabalho mostram que o TAA atinge uma taxa de sucesso mais elevada quando se é utilizada uma perturbação relativamente menor, em comparação com outros métodos de ataque, tais como por exemplo o FGSM e o RP2. Um exemplo da aplicação do TAA num sinal de STOP está ilustrado na Figura 3.5b.

Outro método de ataque que aplica, desta vez, só sombras a um sinal de trânsito é o ataque adversarial desenvolvido pelos autores *Zhong et al.* [47]. Neste trabalho, é estudado mais um tipo de ataque adversarial ótico, em que as perturbações, nesta ocasião, são geradas através do mecanismo *Expectation Over Transformation* (EOT) de forma a se obter uma perturbação naturalista e furtiva num cenário de ataque *Black-Box*. Os dois modelos de rede neuronal utilizados para a obtenção de resultados experimentais relativamente à taxa de erro e precisão são os desenvolvidos no ataque adversarial referenciado inicialmente, isto é, os modelos *LISA*-CNN e GTSRB-CNN do ataque adversarial RP2 treinados nas bases de dados GTSRB e *LISA*. Um exemplo do sinal de cedência de passagem de entrada num cruzamento com via sem prioridade com o ataque desenvolvido pelos autores *Zhong et al.* encontra-se representado na Figura 3.5c.

Por fim, no trabalho desenvolvido pelos autores *Liu et al.* [48], é descrita uma nova abordagem para gerar gotas de chuva nas imagens obtidas por um câmara instalada no veículo. Este ataque adversarial, designado por gotas de chuva adversariais (do inglês, *Adversarial Raindrops* (AdvRD)), utiliza a técnica de redes adversariais generativas (do inglês, *Generative Adversarial Network* (GAN)) de modo a simular gotas de chuva naturais. Como as perturbações de gotas de chuva criadas através do ataque AdvRD são muito semelhantes às gotas de chuva reais e a sua distribuição nas imagens de teste é muito compatível com o caso real, então este tipo de ataque possui uma elevada taxa de erro na classificação de sinais de trânsito para os modelos de DNN de última geração. Por outro lado, também é demonstrado neste trabalho que o treino adversarial adicionando as imagens geradas pelo ataque AdvRD aos dados de treino das bases de dados utilizadas, que são a GTSRB e a TT100K, consegue melhorar significativamente a robustez do modelo. Um exemplo de uma imagem da base de dados TT100K para a classificação e deteção simultânea do sinal de proibição com o ataque desenvolvido pelos autores *Liu et al.* está ilustrado na Figura 3.5d.

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

Figura 3.5: Exemplos de sinais de trânsito manipulados com 4 diferentes ataques adversariais: (a) Ataque RP2 [45]; (b) TAA [46]; (c) Ataque em forma de sombra [47]; (d) Ataque AdvRD [48].

3.2.3 *Classificadores de Sinais de Trânsito*

As redes neuronais convolucionais (CNN) profundas têm sido muito utilizadas na tarefa de classificação de sinais de trânsito (TSC), graças à sua excelente capacidade de extração de características e de obtenção de previsões robustas. Devido a este facto, a maior parte dos classificadores implementados para a tarefa de TSC são baseados em CNN profundas, sendo que a maioria dos resultados experimentais dos trabalhos relacionados com a elaboração desses classificadores centra-se num ou dois aspetos principais, isto é, a precisão ou robustez e o número de parâmetros que o modelo desenvolvido obtém. Assim sendo, seguidamente são referenciados quatro trabalhos de classificadores diferentes para a mesma função de TSC e no subcapítulo 3.2.4 é referenciada e enunciada uma técnica que também realiza o teste de robustez de um ou mais modelos de TSC. Na Tabela 3.5 estão sumariados todos

os 4 trabalhos citados em seguida dos classificadores de sinais de trânsito fazendo a comparação com o seu valor máximo de precisão e o número de parâmetros (quando possível) que cada modelo adquire.

Um dos trabalhos mais antigos de classificadores de sinais de trânsito teve origem no trabalho desenvolvido por *Cireşan et al.*. A abordagem descrita no artigo [49] venceu em 2011 a fase final da competição de classificação de sinais de trânsito, intitulada *German Traffic Sign Recognition Benchmark* [5]. Essa competição, que tem o mesmo nome que a base de dados (GTSRB) [5], consistiu em duas fases, uma avaliação preliminar *online* seguida de uma competição final presencial no local da Conferência Conjunta Internacional sobre Redes Neuronais. O modelo concebido, chamado de rede neuronal profunda de multi-colunas (do inglês, *Multi-Column Deep Neural Network* (MCDNN)), é formado por um total de 25 redes neuronais, sendo que cada 5 redes são treinadas com diferentes métodos (ou filtros) de pré-processamento, o que leva ao aumento da precisão (taxa de reconhecimento) para uma média entre 98.52%-99.46%. Nenhuma das 25 redes neuronais implementadas na MCDNN, individualmente, possui uma taxa de reconhecimento superior com diferentes métodos de pré-processamento, tal como por exemplo a filtragem das imagens da base de dados, mas a sua combinação numa só MCDNN aumenta a robustez do modelo a vários tipos de perturbações/ruído e conduz a um maior número de sinais de trânsito bem classificados. Cada uma das 25 redes da MCDNN possui 9 camadas, sendo três dessas camadas convolucionais, duas de *max pooling*, duas totalmente conectadas, uma de entrada e uma de saída/classificação. A camada de entrada possui a especificação do formato de dados de imagens de 48x48x3 (comprimento x largura x número de canais) e o resultado da saída da MCDNN é calculado através da média do valor na camada final de cada uma das 25 redes neuronais. Relativamente aos resultados experimentais, o modelo implementado venceu a competição GTSRB com uma taxa de reconhecimento (precisão) de sinais de trânsito de 99.46%, sendo superior à taxa de reconhecimento dos seres humanos na tarefa de TSC para condições em que o humano está a dedicar a sua melhor atenção ao problema, *i.e.* as suas melhores condições de atenção (98.84%) [50], e produz três vezes menos erros de classificação do que o segundo melhor algoritmo concorrente (98.31%) [50].

Na sua generalidade, os classificadores mais recentes, têm em conta a executabilidade em tempo real, ou seja, ao longo dos últimos anos foram propostos novos modelos baseado em CNN, eficientes e leves em termos da quantidade de parâmetros, exigindo assim um menor custo computacional, permitindo a sua implementação nos veículos da atualidade.

O classificador para a tarefa de TSC proposto pelos autores *Kherraki et al.* [51] em 2022, consegue resolver o problema de elaboração de um modelo eficiente e

leve, utilizando-se apenas 0.61 milhões de parâmetros, mantendo uma precisão de 96.64%. O modelo de rede neuronal convolucional é constituído por 9 blocos de diferentes camadas, e ao todo, o modelo contém 15 camadas convolucionais com diferentes tamanhos de filtros e de *kernel* entre cada camada convolucional, de modo a se obter um modelo leve em termos do número de parâmetros. Uma grande diferença deste trabalho relativamente aos outros citados em seguida, é que o modelo desenvolvido foi treinado a partir da base de dados BelgiumTS [42] para provar a sua eficácia em outros casos reais de sinais de trânsito sem ser os alemães. Os resultados experimentais, foram obtidos recorrendo a uma CNN profunda treinada ao longo das 200 épocas na base de dados BelgiumTS [42]. Dois algoritmos de otimização de treino foram testados, isto é, os otimizadores *Adaptive Moment Estimation* (Adam) e *Stochastic Gradient Descent* (SGD), e 3 formatos diferentes de dados de entrada, ou seja, 32x32x3, 48x48x3 e 64x64x3. O melhor resultado obtido, tendo em consideração o constante equilíbrio que existe entre a precisão e o número de parâmetros, foi para o formato de dados de entrada de 64x64x3 e o algoritmo de otimização Adam, onde se obteve uma precisão de 96.64% utilizando-se apenas 0.61 milhões de parâmetros, o que torna a dimensão e precisão do modelo desenvolvido adequado para uma aplicação real.

Seguidamente, *Kumar et al.* [52] desenvolveram uma rede neuronal convolucional para classificação de sinais de trânsito, tendo como principal objetivo diminuir o tempo de inferência e também aumentar a precisão global da rede, exigindo assim uma menor potência computacional. A base de dados utilizada para treino, validação e teste do modelo concebido foi a GTSRB [5], e a rede neuronal desenvolvida possui, no total, 14 camadas, tendo 4 camadas convolucionais de diferentes tamanhos de filtro, 3 camadas de *dropout* de modo a evitar o *overfitting* dos dados de treino e 2 camadas totalmente conectadas, sendo que a camada inicial aceita o formato de dados de imagens de resolução 30x30x3. Os resultados experimentais são demonstrados nos valores de precisão e de perda, dos dados de treino e de validação ao longos das 20 épocas de treino, condições às quais o modelo de CNN foi aplicado. Obteve-se um valor máximo de 98.81% de precisão da rede para 0.24 milhões de parâmetros alocados para a tarefa de TSC.

Um dos trabalhos mais recentes no tópico de TSR foi desenvolvido por *Ferencz et al.* [53]. Este trabalho, elaborado em 2024, apresenta uma implementação escolhida de entre várias arquiteturas de algoritmos de reconhecimento e classificação de sinais de trânsito baseadas em CNN. A arquitetura selecionada e treinada na base de dados GTSRB [5] permite classificar 43 tipos diferentes de sinais de trânsito, e a CNN desenvolvida possui 17 camadas no total, 4 camadas convolucionais, 4 camadas de normalização em lote, 2 camadas de *max pooling*, 1 camada de achatamento (*i.e. flatten layer*) , 1 camada de *dropout* e 2 camadas totalmente conectadas. O modelo desenvolvido foi treinado de ponta a ponta durante 20 épocas (número de vezes que

Autores (ano de publicação)	Base(s) de dados	Modelo de rede neuronal	Precisão (%)	Nº de parâmetros (M)
Cireşan et al. (2011) [49]	GTSRB	MCDNN	99.46	-
Kherraki et al. (2022) [51]	BelgiumTS	CNN de 9 blocos	96.64	0.61
Kumar et al. (2022) [52]	GTSRB	CNN de 14 camadas	98.81	0.24
Ferencz et al. (2024) [53]	GTSRB	CNN de 17 camadas	97.98	≈2.7

Tabela 3.5: Visão geral de 4 classificadores de sinais de trânsito.

a rede passa por um processo de treino completo), e os resultados experimentais obtidos indicam que o modelo possui uma precisão máxima de 97.98%, com uma utilização de aproximadamente 2.7 milhões de parâmetros, valor de precisão que está muito próximo do de um ser humano para a mesma tarefa de TSC no seu maior nível de atenção (98.81%) [50].

3.2.4 *Robustez de Classificadores de Sinais de Trânsito*

Berghoff et al. [54] desenvolveram uma abordagem complementar ao desenvolvimento de uma metodologia para testar a robustez de sistemas de AI específicos para a tarefa de TSC. O modelo elaborado pelos autores baseia-se no teste da robustez de diferentes classificadores de sinais de trânsito baseados em DNN com a especificação de determinadas propriedades de robustez. Para a realização do teste de robustez, foram utilizados dois modelos de redes neuronais. O primeiro é uma rede neuronal chamada de *pre-trained* que possui 99.0% de precisão utilizando o modelo de extração de características *Inception_v3* pré-treinado na base de dados *ImageNet*, e depois afinado para a base de dados GTSRB, e o segundo é uma rede neuronal denominada de *self-trained*, que possui uma precisão de 97.4%, e também utiliza o modelo de extração de características *Inception_v3*, mas com uma dimensão reduzida e com nenhum pré-treino. Ambas as redes foram treinadas na base de dados GTSRB com uma política de aumento de dados que aplica algumas transformações aleatórias às imagens, tais como o recorte, a rotação, e alterações de cor de forma a se obter uma uniformização no número de imagens de cada classe da base de dados. Depois disso, as imagens são redimensionadas para a resolução esperada de cada rede, isto é, 32x32 para a rede neuronal *self-trained* e 299x299 para a rede neuronal *pre-trained*. Relativamente às propriedades de robustez, foram definidos 4 tipos diferentes, sendo que numa propriedade podem existir diferentes tipos de ataque:

- Ruído de imagem, tal como o ruído Gaussiano;
- Perturbações de pixeis, utilizando as normas L_0 e L_∞. A norma L_0 permite modificar uma quantidade específica de pixeis numa qualquer extensão e a norma L_∞ permite alterar todos os pixeis tendo em conta um determinado valor limite;
- Transformações geométricas, tal como diferentes pontos de vista, rotação, translação e desfocagem;
- Transformações de cor, tal como alteração do brilho, contraste, saturação, tonalidade e profundidade da cor.

Os resultados experimentais dos testes de robustez realizados pelos autores *Berghoff et al.* [54] estão divididos em testes básicos, onde se realiza a verificação de uma propriedade de robustez e testes de combinações de propriedades, onde se combina duas ou mais propriedades.

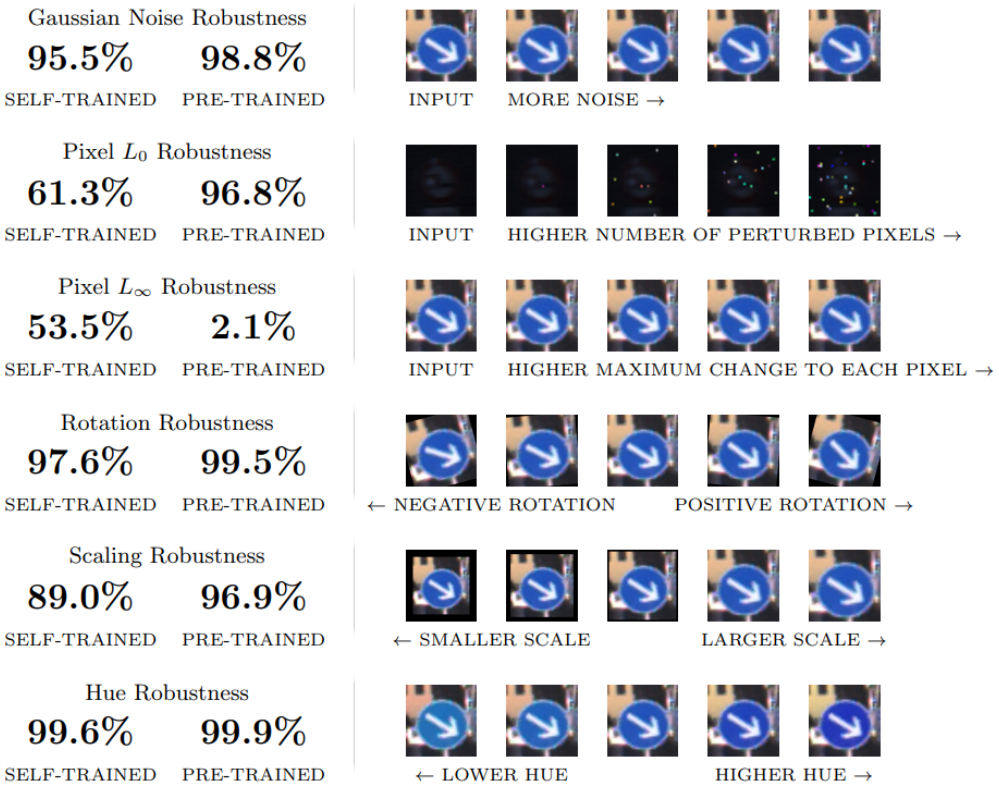

Figura 3.6: Visão geral das pontuações de robustez dos 2 modelos face a diferentes propriedades. Retirada de [54]

Como se pode visualizar na Figura 3.6, a robustez dos dois modelos é calculada em relação às propriedades enunciadas anteriormente, e a pontuação de robustez indica a percentagem de imagens que são bem classificadas (robustas) tendo em conta os ataques aplicados. Da Figura 3.6, também é possível verificar que as pontuações de robustez diferem significativamente, tanto entre cada modelo como entre cada propriedade, e das propriedades selecionadas, a robustez para perturbações de

pixel através da norma L_∞ destaca-se apresentando as pontuações mais baixas, com apenas cerca de metade das classificações robustas para a rede *self-trained* e praticamente nenhuma para a rede *pre-trained*.

No entanto, uma combinação de propriedades de robustez reflete de forma mais realista as condições do mundo real, e por isso, a Figura 3.7 apresenta uma visão geral das pontuações de robustez dos dois modelos quando são testados contra combinações de dois ataques (e propriedades) diferentes. Como se pode analisar, em ambos os casos as pontuações de robustez diminuem, já que por exemplo, a combinação da translação com o dimensionamento da propriedade de robustez de transformações geométricas produz uma pontuação de 85.5%, enquanto a pontuação de robustez apenas para o ataque de translação e de dimensionamento é de 94.7% e 93.0%, respetivamente. Assim, a combinação dos dois ataques diminui a pontuação de robustez da rede *self-trained* de se obter uma classificação robusta em mais 7.5% em comparação com o menor valor das duas pontuações individuais.

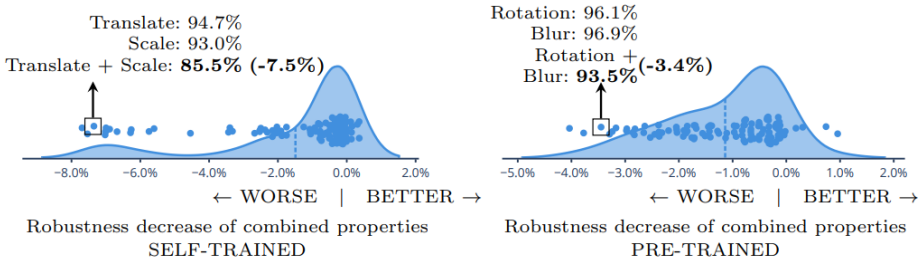

Figura 3.7: Visão geral do efeito da combinação de propriedades de robustez nos 2 modelos em comparação com duas propriedades individuais selecionadas. Retirada de [54]

Este trabalho [54] é muito similar à abordagem detalhada no capítulo 4 desenvolvida no âmbito deste projeto automóvel.

3.3 COMPETIÇÃO INTERNACIONAL DE VERIFICAÇÃO FORMAL DE REDES NEURONAIS

A Competição Internacional de Verificação Formal de Redes Neuronais (VNN-COMP) [55] foi criada de modo a facilitar uma comparação justa e objetiva entre diferentes abordagens e ferramentas de verificação formal de redes neuronais existentes, para reunir a comunidade que trabalha neste problema e ajudar a delinear as futuras direções neste domínio. A VNN-COMP foi estabelecida em 2020, e já se encontra na sua quarta edição. Na última edição, participaram 7 equipas num conjunto diversificado de 10 *benchmarks* com pontuação e 4 *benchmarks* sem pontuação [55].

Como a VNN-COMP trata-se de uma competição, então possui um sistema de pontuação para cada instância (ou teste) realizada. Cada instância é pontuada da seguinte forma [55]:

- A ferramenta afirma que a propriedade é verdadeira e isso de facto é comprovado = 10 pontos;
- A ferramenta informa que uma propriedade é violada e isso é verdadeiro = 10 pontos;
- A ferramenta declara um resultado incorreto (por exemplo, um contra-exemplo não válido) = −150 pontos.

No entanto, o verdadeiro resultado para uma determinada instância não é, geralmente, conhecido a priori. Desta forma, em caso de desacordo entre ferramentas para uma mesma instância, é verificado primeiramente o contra-exemplo considerado pela ferramenta que declara que a propriedade é violada, e caso esse contra-exemplo seja válido, então o resultado correto dessa instância é que a propriedade é violada [55].

A quarta edição da VNN-COMP, relativa ao ano de 2023, continua com a tendência das últimas edições de aumentar a normalização ou padronização de verificação, ou seja, com o objetivo de permitir uma comparação justa entre as ferramentas participantes e simplificar a sua avaliação numa grande variedade de problemas (*benchmarks*) do mundo real. Com esse objetivo em mente, foram definidos três critérios essenciais de participação (ilustrados na Figura 3.8): a utilização do formato *Open Neural Network Exchange* (ONNX) para a especificação de redes neuronais (modelos), o formato *Verified Neural Network Library* (VNN-LIB) para a especificação das propriedades a serem verificadas, e a definição de um limite de tempo máximo (*Timeout*) de 6 horas para a execução da verificação de uma propriedade específica para cada proposta de *benchmark* [55].

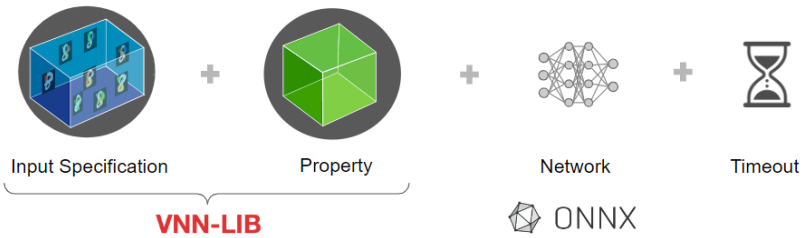

Figura 3.8: Critérios de participação normalizados para cada participante. Retirada de [54]

Ao todo, na VNN-COMP 2023, participaram 7 equipas/ferramentas diferentes, sendo que cada ferramenta pode ter sido construída por um conjunto de duas ou mais universidades, ou organizações. A ferramenta *Neural Network Verification* (NNV) [56], apesar de ter alcançado um escasso 6º lugar em 7 ferramentas, é a que será usada no âmbito do trabalho desta dissertação, uma vez que é a única

baseada em linguagem MATLAB [55]. Como referenciado posteriormente com um maior detalhe no subcapítulo 4.4.2, a ferramenta NNV [56] é um complemento do software MATLAB para a verificação formal de modelos de DL e *Cyber-Physical Systems* (CPS) baseados em redes neuronais. Utiliza uma representação do espaço de dados e de estados num *star set* ou *ImageStar* e um algoritmo de alcançabilidade que permite o cálculo de camada a camada do conjunto alcançável final exato ou sobre-aproximado para redes neuronais *feed-foward* (FFNN), redes neuronais convolucionais (CNN), redes neuronais recorrentes (RNN), bem como sistemas de controlo de redes neuronais (*Neural Network Control Systems* (NNCS)) [55].

Relativamente aos *benchmarks* utilizados, ao longo das quatro edições da competição existiram diversos, mas, mais recentemente, um *benchmark* que foi proposto na edição da VNN-COMP 2022, é o mais adequado para o tipo de estudo relativo à classificação de sinais de trânsito:

- O *benchmark* chamado de *Traffic Signs Recognition* [57], originalmente proposto para a edição da VNN-COMP 2022, procura resolver o problema da suscetibilidade dos sinais de trânsito a exemplos adversariais, decorrentes de diversas condições ambientais ou do quotidiano através do processo de verificação formal. Este *benchmark*, desenvolvido pelos autores *Postovan et al.* [57] baseia-se em redes neuronais binárias (do inglês, *Binary Neural Network* (BNN)), que é sucintamente uma FFNN em que os pesos e as ativações de cada neurónio são valores binários. As BNN, como são redes promissoras para ambientes computacionalmente limitados em termos de memória ou com restrições de energia, e como os pesos e os valores de ativação são binarizados para ± 1, proporcionam um tamanho de modelo reduzido e operações de convolução mais simplificadas para a função de TSR, em comparação com as redes neuronais mais tradicionais, sendo deste modo particularmente pertinentes no domínio da condução autónoma [57]. Para tal propósito, foram disponibilizados três modelos diferentes de BNN elaborados pelos autores, e treinados nas bases de dados GTSRB, BelgiumTS e TT100K ao longo de 30 épocas para seguidamente serem verificados formalmente com a implementação de perturbações da norma L_∞, com os conjuntos de valores limite denotados por $\epsilon = \{1{,}3{,}5{,}10{,}15\}$ [55, 57]. Graças à dificuldade do problema de verificação dada pelo elevado número de parâmetros da rede (905k - 1.7M), da grande dimensão dos dados de entrada (2.7k - 12k) e do número relativamente alto de diferentes classes de sinais de trânsito (43), este *benchmark* não foi incluído no processo de pontuação da edição de 2022 porque depois de algumas conversações com os organizadores e concorrentes, revelou-se que nenhuma ferramenta conseguiria lidar com as especificações do *benchmark*, e além disso, já para a última edição da competição, ou seja, na VNN-COMP 2023, o problema de verificação formal exposto no *benchmark* também foi considerado bastante

complexo porque só 4 das 7 ferramentas é que conseguiram obter resultados meramente credíveis dentro do tempo limite de 6 horas [55].

Esta página foi propositadamente deixada em branco.

4

VFRNP4CA - VERIFICAÇÃO FORMAL DE REDES NEURONAIS PROFUNDAS EM CONTEXTO DE CONDUÇÃO AUTÓNOMA

Neste capítulo é apresentado o trabalho desenvolvido no decorrer deste projeto antes da obtenção dos resultados, relativamente à verificação formal de redes neuronais profundas em contexto de condução autónoma. Inicialmente, é enunciado e explicado o fluxo de trabalho utilizado ao longo do tempo, tendo em consideração a inserção deste tipo de *pipeline* num caso real de aplicação num veículo autónomo. Seguidamente, é clarificado e especificado o conjunto de dados de imagens utilizado durante a realização deste trabalho, isto é, a base de dados, e posteriormente, detalham-se os três modelos de redes neuronais propostos com a intenção de os preparar para serem aplicados em dispositivos de baixa capacidade computacional e de recursos limitados. Por fim, é ilustrado e exemplificado o processo de verificação formal utilizado durante o trabalho, isto é, são especificadas as cinco propriedades de robustez estipuladas e também o software que serviu de apoio para a realização do trabalho, ou seja, o MATLAB, tal como a ferramenta a partir do qual se executou o processo de verificação formal, ou seja, o NNV.

4.1 FLUXO DE TRABALHO

A partir das investigações relevantes enunciadas previamente no Capítulo 3, é possível constatar que, mais recentemente, muitos trabalhos da comunidade de pesquisa, deteção e classificação de sinais de trânsito já começam a ter em consideração o número de parâmetros para além do valor de precisão (ou taxa de classificação) dos modelos de DL e ML desenvolvidos, o que os torna desta maneira mais adequados para aplicações de tempo real. De modo a continuar a abordar este desafio, neste trabalho são propostas três novas redes neuronais convolucionais (CNN) que consigam estabelecer um equilíbrio entre a precisão e o número de parâmetros.

Tendo por base o processo de classificação de sinais de trânsito para um caso real em contexto de condução autónoma, como se verifica na parte superior da Figura 4.1, o procedimento inicia-se com a captação de um frame de vídeo ou imagem de entrada de um sinal de trânsito a partir de uma câmara incorporada na parte frontal do veículo autónomo. Depois, realizam-se as duas tarefas principais do reconhecimento de sinais de trânsito, isto é, a deteção e a classificação. O objetivo deste trabalho

enquadra-se na tarefa de classificação de sinais de trânsito, deste modo na parte inferior da Figura 4.1, encontra-se ilustrado o fluxo de trabalho elaborado por toda a sua extensão. Por último, no diagrama do processo de classificação de sinais de trânsito de um veículo autónomo, obtém-se o rótulo (*label*) do sinal de trânsito correspondente à imagem ou frame de entrada que foi detetado e classificado.

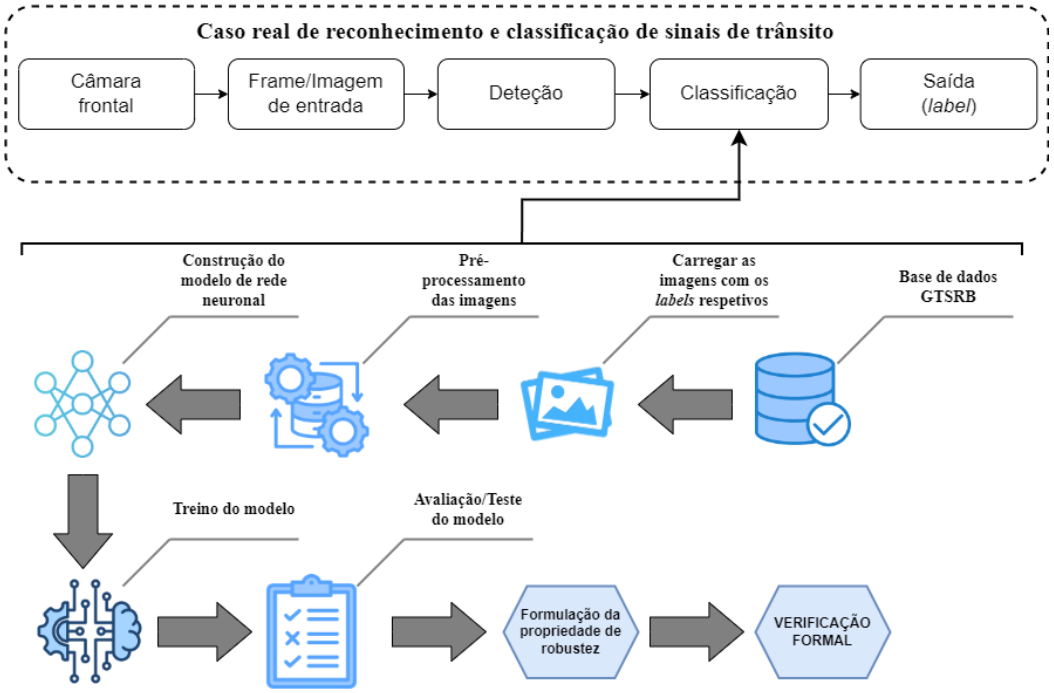

Figura 4.1: Diagrama funcional simplificado e fluxo de trabalho realizado da metodologia de classificação de sinais de trânsito.

Como se pode averiguar na parte inferior da Figura 4.1, há vários processos que devem ser concluídos de modo a se avaliar a robustez dos modelos de CNN propostos. O fluxo de trabalho começa pelo carregamento dos dados de imagens com os *labels* correspondentes a cada imagem a partir da base de dados GTSRB, e pelo pré-processamento dos dados para o tamanho selecionado. Seguidamente, são construídos, e treinados os três modelos diferentes de CNN desenvolvidas no conjunto de dados de treino. Posteriormente ao treino de cada modelo e depois da obtenção do valor respetivo da precisão de validação e da perda de cada CNN, é realizada uma avaliação/teste a cada rede, onde são testadas com 6 diferentes imagens normais do conjunto de teste, isto é, sem qualquer tipo de ataque, e 4 imagens onde são realizados os ataques *White-Box* FGSM e PGD para quatro diferentes valores limite (ϵ) de perturbação. Os dois últimos passos consistem na formulação da propriedade de robustez estipulada com diferentes valores de ϵ definidos, e na verificação formal do modelo de CNN mais simples e com o menor número de parâmetros proposto, através da ferramenta NNV. Deve-se salientar que esta ferramenta foi utilizada durante três (2020, 2021 e 2023) das quatro edições até agora realizadas da VNN-COMP [55].

4.2 BASE DE DADOS

A base de dados GTSRB, que é utilizada durante todo o fluxo de trabalho, consiste num conjunto de dados de imagens amplamente utilizado nas áreas do DL e ML, principalmente para a tarefa de reconhecimento de sinais de trânsito. Esta base de dados, tal como já referido anteriormente no Capítulo 3.2, foi criada como parte de uma competição na Conferência Conjunta Internacional sobre Redes Neuronais de 2011, e projetada de modo a desafiar e aperfeiçoar diferentes algoritmos de reconhecimento de imagens baseados em redes neuronais profundas, em especial os que são desenvolvidos em contexto de condução autónoma ou assistência ao condutor [5].

A base de dados GTSRB contém, ao todo, mais de 50000 imagens de sinais de trânsito, que são distribuídas entre 43 classes diferentes, tais como sinais de limite de velocidade, sinais de advertência, sinais de obrigação e sinais de proibição. Na Figura 4.2, é demonstrado um exemplo de imagem em sequência crescente de classe, ou seja, da classe 0 até a classe 42 para cada uma das 43 classes de sinais de trânsito presentes na base de dados GTSRB. As imagens foram retiradas de diversos cenários da Alemanha, o que oferece uma variedade significativa de condições de iluminação, distâncias e ângulos de captura, tornando assim, a tarefa de classificação dos sinais de trânsito mais desafiadora porque inclui imagens de baixa qualidade, distorções, oclusões e sinais parcialmente visíveis [5].

Figura 4.2: Exemplos de imagens de sinais de trânsito retirados da base de dados GTSRB, um para cada classe. Retirada de [5]

O conjunto de dados é dividido em dois grandes grupos:

- Treino + Validação: 39209 imagens a cores (do tipo RGB) que são utilizadas para o treino e validação dos modelos de reconhecimento;
- Teste: 12630 imagens a cores (do tipo RGB) que são usadas para avaliar o desempenho dos modelos treinados.

Cada imagem no conjunto de dados, tanto de treino como de teste, é complementada com informação da classe (ou *label*) do sinal de trânsito, e a resolução especificada em altura e largura, sendo que esses valores variam entre 25 a 266 e 25 a 232 pixeis, respetivamente.

Treino + Validação		**Teste**	
Rótulos	**# imagens**	**Rótulos**	**# imagens**
0 e 1	210 e 2220	0 e 1	60 e 720
2 e 3	2250 e 1410	2 e 3	750 e 450
4 e 5	1980 e 1860	4 e 5	660 e 630
6 e 7	420 e 1440	6 e 7	150 e 450
8 e 9	1410 e 1470	8 e 9	450 e 480
10 e 11	2010 e 1320	10 e 11	660 e 420
12 e 13	2100 e 2160	12 e 13	690 e 720
14 e 15	780 e 630	14 e 15	270 e 210
16 e 17	420 e 1110	16 e 17	150 e 360
18 e 19	1200 e 210	18 e 19	390 e 60
20 e 21	360 e 330	20 e 21	90 e 90
22 e 23	390 e 510	22 e 23	120 e 150
24 e 25	270 e 1500	24 e 25	90 e 480
26 e 27	600 e 240	26 e 27	180 e 60
28 e 29	540 e 270	28 e 29	150 e 90
30 e 31	450 e 780	30 e 31	150 e 270
32 e 33	240 e 689	32 e 33	60 e 210
34 e 35	420 e 1200	34 e 35	120 e 390
36 e 37	390 e 210	36 e 37	120 e 60
38 e 39	2070 e 300	38 e 39	690 e 90
40 e 41	360 e 240	40 e 41	90 e 60
42	240	42	90
Total	**39209**	**Total**	**12630**

Tabela 4.1: Dados de treino, validação e de teste da base de dados GTSRB.

Apesar de ser uma das bases de dados mais utilizadas no domínio de classificação de sinais de trânsito para modelos de DL e ML, como se pode apurar a partir da Tabela 4.1 e das Figuras 4.3 e 4.4, uma das principais desvantagens da base de dados GTSRB situa-se no grande desequilíbrio que existe na distribuição de imagens para cada classe do conjunto de dados, tanto de treino como de teste.

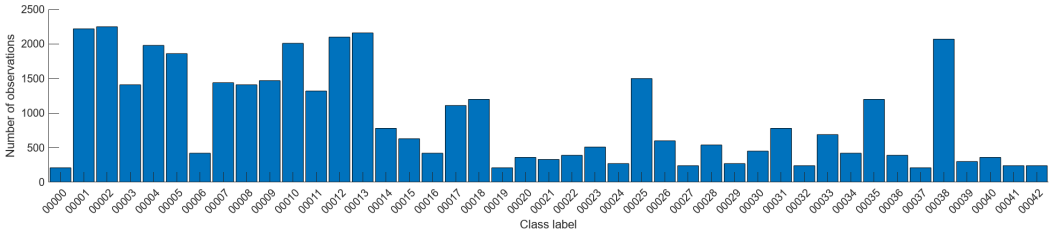

Figura 4.3: Número de imagens por cada classe do conjunto de dados de treino e validação da base de dados GTSRB.

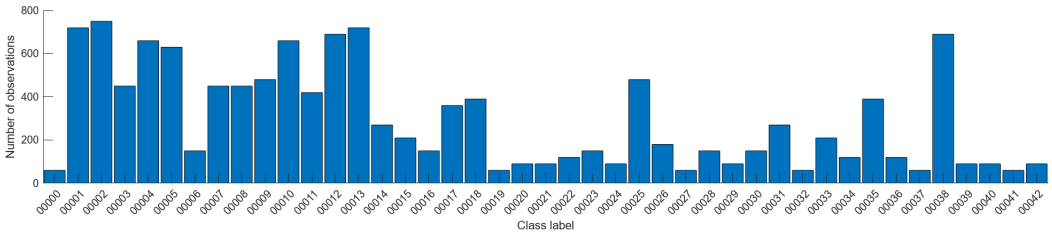

Figura 4.4: Número de imagens por cada classe do conjunto de dados de teste da base de dados GTSRB.

Esta desproporção do número de observações entre as classes afeta negativamente o desempenho dos modelos ou algoritmos que são treinados e testados através da base de dados, pois as redes neuronais depois do processo de treino tendem a ter uma maior precisão na classificação de sinais de trânsito das classes maioritárias, enquanto apresentam dificuldades em obter uma classificação correta e precisa dos sinais de trânsito pertencentes às classes minoritárias. Como resultado, o modelo que é treinado e testado a partir da base de dados GTSRB deve ser cuidadosamente avaliado, principalmente através do conjunto de dados de validação, de modo a não ficar *overfitted*, isto é, com ajuste excessivo ao conjunto de dados de treino, favorecendo as classes de sinais de trânsito com mais imagens e prejudicando a generalização para classes com uma menor porção de imagens de sinais de trânsito.

4.3 MODELOS DE REDES NEURONAIS

Três CNN foram desenvolvidas tendo como objetivo minimizar os recursos em termos de número de parâmetros, mantendo uma elevada taxa de classificação (precisão). Estes três modelos servirão para proceder ao estudo do problema da obtenção de redes neuronais profundas robustas e precisas na tarefa de classificação de sinais

CNN 1		CNN 2		CNN 3	
Camadas	**Detalhes**	**Camadas**	**Detalhes**	**Camadas**	**Detalhes**
ImageInput	tamanho: [30,30,3]	*ImageInput*	tamanho: [32,32,3]	*ImageInput*	tamanho: [48,48,3]
Conv2D	filtros: 8, tamanho de kernel: 3	*Conv2D*	filtros: 16, tamanho de kernel: 3	*Conv2D*	filtros: 16, tamanho de kernel: 3
ReLU	-	*ReLU*	-	*ReLU*	-
BatchNorm	tamanho: 8	*BatchNorm*	tamanho: 16	*BatchNorm*	tamanho: 16
MaxPooling2D	tamanho de pool: 2	*MaxPooling2D*	tamanho de pool: 2	*MaxPooling2D*	tamanho de pool: 2
Conv2D	filtros: 16, tamanho de kernel: 3	*Conv2D*	filtros: 32, tamanho de kernel: 3	*Conv2D*	filtros: 32, tamanho de kernel: 3
ReLU	-	*ReLU*	-	*ReLU*	-
BatchNorm	tamanho: 16	*BatchNorm*	tamanho: 32	*BatchNorm*	tamanho: 32
MaxPooling2D	tamanho de pool: 2	*MaxPooling2D*	tamanho de pool: 2	*MaxPooling2D*	tamanho de pool: 2
		Conv2D	filtros: 64, tamanho de kernel: 3	*Conv2D*	filtros: 64, tamanho de kernel: 3
FC	neurónios: 43	*ReLU*	-	*ReLU*	-
Softmax	-	*BatchNorm*	tamanho: 64	*BatchNorm*	tamanho: 64
ClassOutput	neurónios: 43 (rótulos)	*MaxPooling2D*	tamanho de pool: 2	*MaxPooling2D*	tamanho de pool: 2
		FC	neurónios: 43	*FC*	neurónios: 43
		Softmax	-	*Softmax*	-
		ClassOutput	neurónios: 43 (rótulos)	*ClassOutput*	neurónios: 43 (rótulos)

Tabela 4.2: Operações utilizadas em cada camada das três redes neuronais convolucionais propostas.

de trânsito. Simultaneamente, os três modelos devem ter um pequeno número de parâmetros de modo a ser possível a sua implementação em casos práticos reais em contexto da condução autónoma (ou seja, veículos autónomos).

Os três modelos de classificação são constituídos por duas ou três camadas convolucionais que são responsáveis pela extração de recursos/características e uma camada totalmente conectada responsável pela atribuição dos rótulos (*labels*) de cada sinal de trânsito. As Figuras 4.5, 4.6 e 4.7, e a Tabela 4.2 detalham minuciosamente toda a estrutura de cada uma das três CNN propostas (*CNN 1*, *CNN 2* e *CNN 3*), bem como as operações utilizadas em cada camada. Enquanto as camadas convolucionais são compostas essencialmente por operações de convolução (*Conv2d*), normalização em lote (*BatchNorm*), *max pooling* (*MaxPooling2D*) e uma função de ativação *ReLU*, a camada totalmente conectada é composta essencialmente por uma camada densa (*FC*), sendo que a classificação é realizada através dos valores dessa camada com a função de ativação *Softmax* de modo a se obter uma previsão num formato de probabilidade para a camada de classificação final (*ClassOutput*).

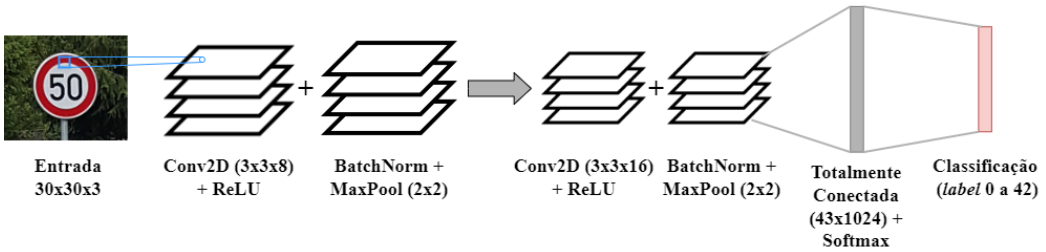

Figura 4.5: Esquema da estrutura da rede neuronal convolucional CNN 1, com 2 camadas convolucionais e especificação de entrada 30x30x3.

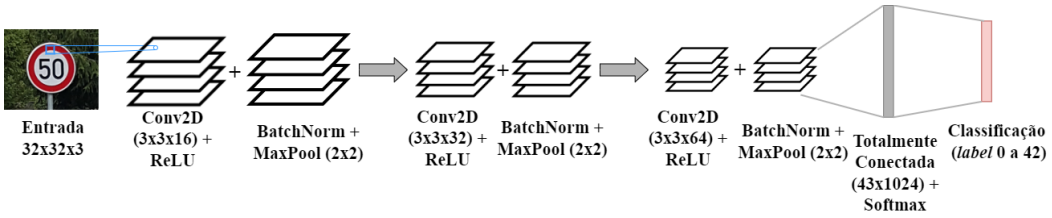

Figura 4.6: Esquema da estrutura da rede neuronal convolucional CNN 2, com 3 camadas convolucionais e especificação de entrada 32x32x3.

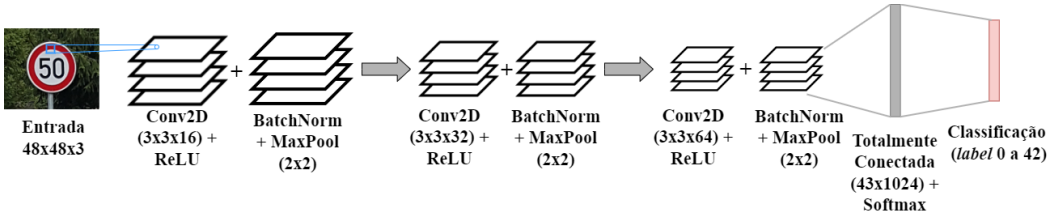

Figura 4.7: Esquema da estrutura da rede neuronal convolucional CNN 3, com 3 camadas convolucionais e especificação de entrada 48x48x3.

Como se é possível observar através da tabela e das três figuras anteriores, o modelo *CNN 1* é composto por duas camadas convolucionais com tamanho de *kernel* igual a 2 para cada camada e diferentes tamanhos de filtro de 8 e 16, respetivamente, e os modelos *CNN 2* e *CNN 3* são compostos por três camadas convolucionais com tamanho de *kernel* igual a 2 para cada camada e diferentes tamanhos de filtro de 16, 32 e 64, respetivamente. Após cada camada convolucional dos três modelos, são adicionadas duas camadas de normalização em lote e *ReLU* e uma camada de *max pooling* com um tamanho de *pool* de 2, a fim de reduzir os mapas de características e o tempo de computação. Depois, em cada um dos três modelos, também é aplicada uma camada totalmente conectada de 43 neurónios que reflete o número total de *labels* das classes dos sinais de trânsito, para finalmente, ser utilizada a camada *Softmax* de modo a normalizar os valores de probabilidade de cada *label* para a camada de classificação final.

As maiores diferenças entre as três redes desenvolvidas (*CNN 1*, *CNN 2* e *CNN 3*) baseiam-se na dimensão aceite para o conjunto de dados de imagens de entrada, no número de camadas convolucionais e no número de filtros de cada camada convolucional. Relativamente à grandeza aceite para as imagens de entrada, optou-se por desenvolver três CNN com tamanhos diferentes de modo a se testar a influência desse parâmetro no número de parâmetros da rede, sendo que, por exemplo, uma imagem de entrada com tamanho 30x30x3 significa que é uma matriz multidimensional que contém valores de pixeis, tal como uma matriz tradicional, com uma largura de 30 (número de colunas), uma altura de 30 (número de linhas), e uma profundidade ou número de canais de 3, o que neste caso significa que é uma imagem RGB. Relativamente ao número de camadas convolucionais e número de filtros em cada camada convolucional, depois de se conceber a rede *CNN 1*, também

se decidiu criar duas CNN mais profundas e complexas (*CNN 2* e *CNN 3*) de forma a se analisar a variação do valor da precisão de classificação de cada rede tendo em conta uma extração de características mais completa e ampla, já que com um maior número de camadas convolucionais e de filtros, os dois modelos *CNN 2* e *CNN 3* conseguem extrair atributos e características da imagem de entrada mais complexos que o modelo *CNN 1*.

4.4 VERIFICAÇÃO FORMAL

O processo de verificação formal usado de forma a garantir que o modelo treinado para a classificação de sinais de trânsito tem em consideração as especificações das propriedades de robustez desejadas é ilustrado na Figura 4.8.

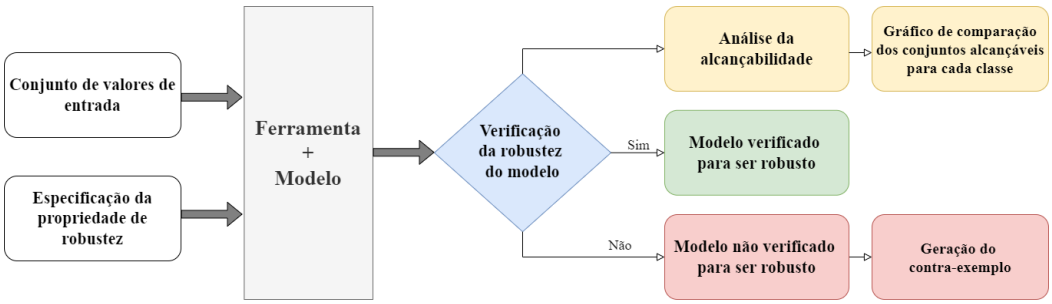

Figura 4.8: Procedimento de verificação formal da propriedade de robustez de um modelo utilizado no trabalho.

Como se analisa a partir da Figura 4.8, no processo de verificação utilizado para este trabalho são consideradas duas fontes de entrada de dados, isto é, o "conjunto de valores de entrada", que representa a agregação dos valores de pixeis de uma imagem RGB representativa de um sinal de trânsito, e a "propriedade de robustez" a ser verificada, ou seja, qual o requisito formal que o modelo deve satisfazer para uma determinada perturbação. A partir destas duas fontes de entradas, a "ferramenta e o modelo" utilizados irão proceder à analise do comportamento dos valores de entrada tendo em conta o intervalo de perturbação designado na propriedade de robustez. No fim do processo de verificação formal, a ferramenta gera dois tipos de resultados, isto é, um "gráfico de comparação dos conjuntos alcançáveis finais de cada classe" como análise da alcançabilidade do modelo, e uma mensagem a indicar se a "verificação do modelo foi bem sucedida", ou se a "verificação do modelo não foi bem sucedida", sendo que nesse caso a ferramenta também "gera um contra-exemplo" que demonstra uma situação onde o modelo utilizado falha no cumprimento da propriedade de robustez.

Apesar de não estar ilustrado na Figura 4.8, tal como acontece todos os anos na VNN-COMP e como referenciado no capítulo 3.3, caso o processo de verificação

formal usado não conseguir gerar a mensagem a indicar que o modelo é robusto, ou a indicar que o modelo não é robusto, até um determinado intervalo de tempo, então para esse caso é admitido que a ferramenta não consegue validar ou comprovar a robustez do modelo para a propriedade de robustez estipulada, e é originado o resultado "Timeout". De modo a aproximar este trabalho com a edição da competição internacional de verificação de redes neuronais (VNN-COMP) de 2023, o máximo intervalo de tempo considerado para a ferramenta utilizada durante a verificação formal do modelo selecionado é 6 horas, ou seja, 21600 segundos.

4.4.1 *Propriedades de Robustez*

A verificação formal é aplicada a propriedades que relacionam os valores de entrada e de saída, onde se define um intervalo para cada valor de entrada e se estabelece uma verificação aplicada aos valores de saída.

Para o caso específico de classificação de imagens de sinais de trânsito, as propriedades de robustez são aplicadas ao conjunto de valores de pixeis de cada imagem a ser testada, ou seja, cada valor de um pixel presente na matriz multidimensional representativa de uma imagem RGB de um sinal de trânsito vai ser alterado, através de um intervalo de perturbações globais determinadas pelo valor limite definido como ϵ. Desta forma, cada valor de um pixel da imagem irá ser modificado entre um intervalo de valores de perturbações globais estipulado como $[-\epsilon, +\epsilon]$. Este tipo de ataque não é baseado no uso do gradiente para se obter o melhor valor de perda e ser o mais eficiente possível no ataque, como os ataques FGSM e PGD, mas sim na perturbação adversarial resultante do cálculo da função de norma L_∞ globalmente.

Neste trabalho foram estipuladas, ao todo, 5 propriedades de robustez, sendo que todas as perturbações estabelecidas são do tipo global e são calculadas utilizando a norma L_∞, onde a única diferença situa-se nos diferentes valores limite (ϵ) especificados. O conjunto dos 5 valores limite de perturbação L_∞ estipulados são $\epsilon = \{0.1, 0.5, 1, 2, 3\}$. Assim, tendo como base a Equação 2.4 (Definição 2.1) do capítulo 2.2.2, as 5 definições das propriedades de robustez utilizadas para a obtenção de resultados neste trabalho são:

DEFINIÇÃO 4.1 - Propriedade de Robustez contra ataque adversarial através da norma L_∞ e $\epsilon = 0.1$. *Dado um modelo de rede neuronal que mapeia um conjunto de valores de pixeis de entrada x_a até um label de saída y_a e, além disso, produz um label de saída y_b dado uma entrada adversarial x_b. E seja $||\cdot||_\infty$ a função para calcular o valor da norma L_∞, então a propriedade de*

robustez ϕ_1 contra qualquer perturbação adversarial no âmbito de 0.1 *é definida da seguinte forma:*

$$\phi_1(x_a,y_a,x_b,y_b,0.1) = (||x_a - x_b||_\infty \leqslant 0.1) \rightarrow (y_a = y_b) \tag{4.1}$$

DEFINIÇÃO 4.2 - Propriedade de Robustez contra ataque adversarial através da norma L_∞ e $\epsilon = 0.5$. *Dado um modelo de rede neuronal que mapeia um conjunto de valores de pixeis de entrada x_a até um label de saída y_a e, além disso, produz um label de saída y_b dada uma entrada adversarial x_b. E seja $||\cdot||_\infty$ a função para calcular o valor da norma L_∞, então a propriedade de robustez ϕ_2 contra qualquer perturbação adversarial no âmbito de* 0.5 *é definida da seguinte forma:*

$$\phi_2(x_a,y_a,x_b,y_b,0.5) = (||x_a - x_b||_\infty \leqslant 0.5) \rightarrow (y_a = y_b) \tag{4.2}$$

DEFINIÇÃO 4.3 - Propriedade de Robustez contra ataque adversarial através da norma L_∞ e $\epsilon = 1$. *Dado um modelo de rede neuronal que mapeia um conjunto de valores de pixeis de entrada x_a até um label de saída y_a e, além disso, produz um label de saída y_b dada uma entrada adversarial x_b. E seja $||\cdot||_\infty$ a função para calcular o valor da norma L_∞, então a propriedade de robustez ϕ_3 contra qualquer perturbação adversarial no âmbito de* 1 *é definida da seguinte forma:*

$$\phi_3(x_a,y_a,x_b,y_b,1) = (||x_a - x_b||_\infty \leqslant 1) \rightarrow (y_a = y_b) \tag{4.3}$$

DEFINIÇÃO 4.4 - Propriedade de Robustez contra ataque adversarial através da norma L_∞ e $\epsilon = 2$. *Dado um modelo de rede neuronal que mapeia um conjunto de valores de pixeis de entrada x_a até um label de saída y_a e, além disso, produz um label de saída y_b dada uma entrada adversarial x_b. E seja $||\cdot||_\infty$ a função para calcular o valor da norma L_∞, então a propriedade de robustez ϕ_4 contra qualquer perturbação adversarial no âmbito de* 2 *é definida da seguinte forma:*

$$\phi_4(x_a,y_a,x_b,y_b,2) = (||x_a - x_b||_\infty \leqslant 2) \rightarrow (y_a = y_b) \tag{4.4}$$

DEFINIÇÃO 4.5 - Propriedade de Robustez contra ataque adversarial através da norma L_∞ e $\epsilon = 3$. *Dado um modelo de rede neuronal que mapeia um conjunto de valores de pixeis de entrada x_a até um label de saída y_a e, além disso, produz um label de saída y_b dada uma entrada adversarial x_b. E seja $||\cdot||_\infty$ a função para calcular o valor da norma L_∞, então a propriedade de robustez ϕ_5 contra qualquer perturbação adversarial no âmbito de* 3 *é definida da seguinte forma:*

$$\phi_5(x_a,y_a,x_b,y_b,3) = (||x_a - x_b||_\infty \leqslant 3) \rightarrow (y_a = y_b) \tag{4.5}$$

4.4.2 *Ferramentas de Verificação Formal*

Das ferramentas desenvolvidas para a verificação formal, nomeadamente as descritas no estado da arte e no capítulo 3.3 relativo à competição internacional de verificação de redes neuronais (VNN-COMP), aquela que é mais pertinente para este trabalho é a NNV porque é uma das únicas compatível e suportada pelo software (ou linguagem) MATLAB. Deste modo, será esta a ferramenta utilizada para a verificação formal de um modelo de rede neuronal convolucional (CNN) selecionado com o intuito de comprovar ou validar as propriedades de robustez estipuladas anteriormente.

Seguidamente, é realizado um texto introdutório sobre como é que o problema deste trabalho se incorpora no software MATLAB, e por fim, é clarificado o modo de funcionamento da ferramenta NNV, onde se é esclarecido, principalmente, o método de análise de alcançabilidade utilizado, isto é, o *ImageStar*, de tal modo que este problema de verificação é delineado como um problema de alcançabilidade.

MATLAB

O MATLAB [4] é uma poderosa ferramenta computacional amplamente utilizada em vários domínios, incluindo a verificação de modelos de redes neuronais. Este será utilizado em particular no que se refere a tarefas de classificação de imagens, tais como a classificação de sinais de trânsito. O MATLAB oferece um vasto conjunto de ferramentas e funções adaptadas para conceber, treinar e verificar formalmente redes neuronais, sendo que um dos seus pontos fortes principais reside no ambiente abrangente que integra a preparação de dados, o desenvolvimento de modelos e o teste desses mesmos modelos num fluxo de trabalho unificado. Em classificação de imagens, o MATLAB também facilita o pré-processamento dos dados de imagem, já que inclui opções como o redimensionamento, normalização e amplificação, o que ajuda no melhoramento da robustez de uma rede neuronal.

No contexto da verificação, o MATLAB, disponibiliza ferramentas como o *Deep Network Designer* [58] e a *Model Verification Library*. Estas ferramentas possibilitam o teste e a validação sistemáticas de redes neuronais, e também permitem o treino de vários cenários de modo a garantir que o modelo de rede neuronal concebido consegue classificar robustamente os sinais de trânsito em diversas condições, incluindo por exemplo, variações de iluminação, oclusões e distorções.

No âmbito deste trabalho apenas será utilizada a ferramenta *Deep Network Designer* (que é explicada no Anexo A) para a criação e treino das redes neuronais construídas, uma vez que para a verificação será usada a ferramenta NNV.

NNV

A ferramenta NNV [56] é uma *toolbox* orientada para objetos e funções escritas em linguagem MATLAB desenvolvida pelas universidades de *Vanderbilt* e de *Nebraska-*

Lincoln dos Estados Unidos da América. É uma estrutura complementar de software que efetua uma verificação baseada em conjuntos para redes neuronais profundas (DNN) e sistemas ciber-físicos (CPS), sendo que estes também são conhecidos como sistemas de controlo de redes neuronais (NNCS).

A NNV fornece um vasto conjunto de algoritmos de alcançabilidade que conseguem calcular ambos os conjuntos exatos ou sobre-aproximados de alcançabilidade de redes neuronais profundas e de sistemas de controlo de redes neuronais usando uma variedade de representações de conjuntos, tais como poliedros, conjuntos de estrela (ou *star-sets*) [59], zonótopos e representações abstratas de domínios [60]. O conjunto alcançável de saída obtido a partir da ferramenta NNV contém todos os estados possíveis de uma DNN a partir dos conjuntos de entrada definidos, ou de um NNCS a partir dos conjuntos de estados iniciais [56].

No que diz respeito à verificação, a NNV declara que uma rede neuronal profunda é segura e robusta se, e só se, os seus conjuntos alcançáveis de saída não violam as propriedades de robustez (ou seja, se existe alguma interseção com qualquer estado que satisfaça a negação da propriedade de robustez). Caso alguma propriedade de robustez seja violada, a ferramenta NNV pode construir um conjunto completo de contra-exemplos que demonstra o conjunto de todas as entradas e estados iniciais (para o caso de NNCS) possíveis que não são seguras e robustas.

Características

A ferramenta NNV possui 2 módulos principais, isto é, um motor de cálculo (*Computation Engine*) e um analisador (*Analyzer*), como estão apresentados na Figura 4.9 [56].

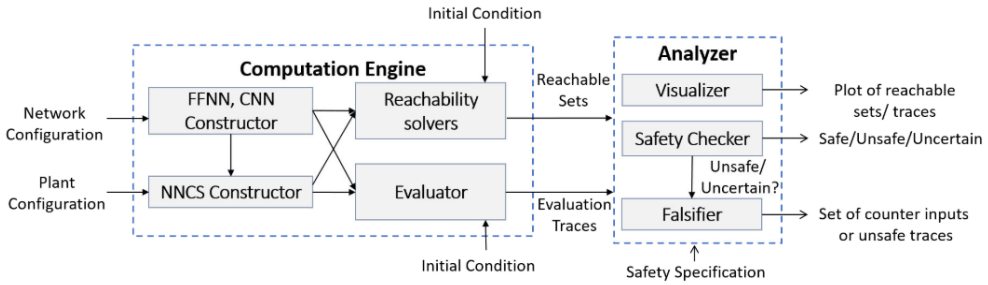

Figura 4.9: Visão geral da ferramenta NNV e dos seus módulos e componentes principais. Retirada de [56]

O módulo do motor de cálculo é composto por 4 subcomponentes, ou seja, o construtor FFNN ou CNN (*FFNN, CNN Constructor*), dependendo do tipo de rede utilizada, o construtor NNCS (*NNCS Constructor*), os solvers de alcançabilidade (*Reachability solvers*), e por último o avaliador (*Evaluator*). Enquanto o construtor FFNN (ou CNN) recebe como entrada um ficheiro de configuração do tipo específico de rede e gera um objeto desse tipo de rede, o construtor NNCS irá receber o objeto mencionado anteriormente e a configuração de planta, que descreve a dinâmica de

Funcionalidade	Ánalise Exata	Análise Sobre-Aproximada
Componentes	FFNN, CNN, NNCS	FFNN, CNN, NNCS
Funções de ativação	ReLU	ReLU, Sigmóide, Tanh
Métodos de alcançabilidade	Star, Poliedra, ImageStar	Star, Zonótopo, Interpretação abstrata, ImageStar
Verificação de robustez (para FFNN/CNN)	Sim	Sim
Falsificação	Sim	Sim
Geração de contra-exemplos	Sim	Sim

Tabela 4.3: Visão geral das principais características da ferramenta NNV. Adaptada de [56]

um sistema, como entradas e, em seguida, cria um objeto do tipo NNCS. Dependendo da aplicação do modelo em estudo, o objeto criado será introduzido num solver de alcançabilidade de forma a calcular o conjunto alcançável do modelo a partir de um determinado conjunto inicial de entradas ou estados.

Seguidamente, o conjunto alcançável obtido pelo módulo do motor de cálculo será passado para o módulo do analisador. O módulo do analisador é composto por 3 sub-componentes, isto é, o visualizador (*Visualizer*), o verificador de segurança/robustez (*Safety Checker*), e o falsificador (*Falsifier*). O visualizador é utilizado para representar, ou desenhar o conjunto alcançável, e dada uma especificação (propriedade) de segurança/robustez, o verificador analisa a possível robustez do modelo em relação à especificação (propriedade). Quando se é utilizado um solver de alcançabilidade exato (sólido e completo), o verificador ou retorna "*safe*", ou "*unsafe*", juntamente com um conjunto de contra-exemplos, e quando se utiliza um solver de alcançabilidade sobre-aproximado (sólido), o verificador ou devolve "*safe*", ou "*uncertain*" (não consegue comprovar a segurança, ou robustez do modelo). Neste caso, o falsificador chama automaticamente o avaliador de forma a produzir traços de avaliação para encontrar um contra-exemplo. No caso do falsificador encontrar um contra-exemplo, a ferramenta NNV retorna "*unsafe*".

Na Tabela 4.3 são resumidas as principais características da ferramenta NNV.

ImageStar

No domínio da condução autónoma, uma classificação precisa e fiável dos diferentes sinais de trânsito existentes nas vias ou estradas públicas é fundamental de forma a garantir uma navegação em segurança. A verificação formal de modelos de redes neuronais utilizadas em tarefas de classificação de imagens surge assim, como um processo vital para garantir a coerência e a robustez destes sistemas, sendo que uma das abordagens mais recentes e avançadas neste domínio é a utilização do algoritmo de alcançabilidade, integrado na *toolbox Neural Network Verification* (NNV) do MATLAB, chamado *ImageStar* [61].

O *ImageStar* é uma nova forma de estrutura de dados e um método de análise de alcançabilidade concebido para processar conjuntos de imagens e transformações de grande dimensão. Representa conjuntos de imagens com várias perturbações de forma eficiente, o que possibilita a realização de tarefas de verificação formal que são computacionalmente complexas e tradicionalmente pouco viáveis. Os principais componentes e conceitos do *ImageStar* são [61]:

- Representação *ImageStar*: Representa uma imagem e as suas perturbações aplicadas baseado num *star set*, que é uma combinação de uma imagem base, restrições de perturbação e parâmetros de transformação afim;
- Transformação camada a camada: O conjunto alcançável de saída de uma rede neuronal, a partir da formulação do problema de alcançabilidade, é construído camada a camada. O cálculo central do conjunto alcançável é realizado em cada camada definida por uma operação específica, isto é, convolução, mapeamento afim, *max pooling*, *average pooling*, ou funções de ativação não lineares tipo ReLU;
- Análise de alcançabilidade: Calcula o conjunto alcançável de todas as saídas para um determinado *ImageStar* de entrada, abrangendo todas as saídas possíveis do modelo de rede neuronal para as perturbações definidas, sendo a partir da análise desses conjuntos que se realiza a verificação da propriedade de robustez.

A definição do método de alcançabilidade *ImageStar*, numa linguagem matemática, é enunciada em seguida:

DEFINIÇÃO 4.6 - *Um ImageStar* Θ é um conjunto $< c,V,P >$ *onde* $c \in \mathbb{R}^{h \times w \times nc}$ *é a imagem de referência,* $V = \{v_1, v_2, \ldots, v_m\}$ *é o conjunto de m imagens em* $\mathbb{R}^{h \times w \times nc}$ *chamadas imagens geradas,* $P : \mathbb{R}^m \rightarrow \{\top,\bot\}$ *é o predicativo, e h, w, nc são, respetivamente, a altura, largura e número de canais das imagens. As imagens geradas são dispostas de modo a formar uma matriz do tipo ImageStar de base* $h \times w \times nc \times m$. *O conjunto de imagens representadas pelo ImageStar é dado por:*

$$\Theta = \{x \mid x = c + \sum_{i=1}^{m} (\alpha_i v_i) \text{ onde } P(\alpha_i, \ldots, \alpha_m) = \top\}. \tag{4.6}$$

Dando como exemplo, uma imagem em tons de cinza 4x4x1 (altura x largura x número de canais) com um limite inferior e superior de perturbação $\epsilon \in [-2,2]$ aplicada no pixel de posição $\{1,2,1\}$, a imagem pode ser descrita como uma representação *ImageStar*, tal como se demonstra na Figura 4.10.

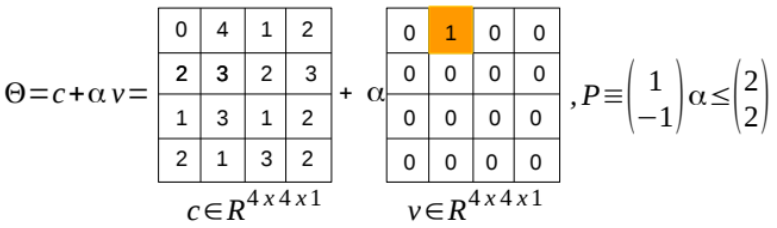

Figura 4.10: Exemplo de uma representação *ImageStar* de uma imagem em tons de cinza com perturbação. Retirada de [61]

Como a representação *ImageStar* pode ser usada para representar um conjunto de imagens distorcidas por um ataque adversarial realizado, então de forma a construir conjuntos alcançáveis de saídas que contêm todos os resultados possíveis de uma rede neuronal, desenvolve-se algoritmos de alcançabilidade *ImageStar* exatos e sobre-aproximados.

Para o processo de cálculo dos conjuntos de alcançabilidade de camada a camada, tipicamente, as camadas finais de um modelo de rede neuronal, como por exemplo, as camadas da função de ativação *Softmax* e de classificação, não são contabilizadas porque não efetuam nenhuma alteração nos resultados obtidos. A camada convolucional, a totalmente conectada, a de normalização em lote e a de *average pooling* são fáceis de entender a sua lógica, porque o processo de cálculo dos conjuntos de alcançabilidade é igual ao seu método base. No que se refere às camadas de *max pooling* e da função de ativação ReLU, é exigido um maior nível de prudência porque os processos de cálculo são diferentes para os diferentes tipos de algoritmos, ou seja, para o caso do algoritmo de alcançabilidade exato existe uma forma e para o caso sobre-aproximado existe outra [61].

Esta página foi propositadamente deixada em branco.

5

ENSAIOS E RESULTADOS EXPERIMENTAIS

Neste capítulo, serão apresentados os resultados e sua interpretação, especificamente serão apresentados os resultados experimentais e as suas respetivas argumentações ou discussões.

Os resultados obtidos podem ser divididos em quatro categorias:

- Resultados relativos ao treino e validação das 3 diferentes CNN desenvolvidas e especificadas no capítulo 4.3, onde se adquiriu o valor de precisão, de perda mínima e do número de parâmetros para cada modelo;
- Resultados relativamente ao teste dos 3 modelos de CNN desenvolvidas com 6 imagens aleatoriamente escolhidas do conjunto de dados de teste sem nenhum ataque, onde se obteve o nível de confiança de cada classificação e o *label* classificado para cada imagem;
- Resultados relacionados ao teste de apenas um dos 3 modelos de CNN desenvolvidas, com 4 imagens aleatoriamente escolhidas do conjunto de dados de teste da base de dados GTSRB, onde se aplicou 2 tipos de ataques *White-Box* baseados no cálculo do gradiente, denominados de FGSM e PGD, e se adquiriu o exemplo adversarial, a ilustração da perturbação e o *label* classificado para cada uma das 4 imagens;
- Resultados relativos à verificação formal na ferramenta NNV do mesmo modelo de CNN e também as mesmas 4 imagens utilizadas previamente para a aplicação dos ataques FGSM e PGD, de modo a comprovar formalmente os resultados obtidos para cada ataque, já que se realizou a análise da alcançabilidade do modelo através do método *ImageStar* para as 5 propriedades de robustez estipuladas no capítulo 4.4.1.

Desta forma, inicialmente, o foco deste capítulo recai sobre os resultados do processo de aprendizagem de treino e de validação das três CNN criadas (capítulo 5.1). Seguidamente, o foco altera-se para os resultados do conjunto de teste a partir da aplicação de dois ataques adversariais com cinco valores limites (ϵ) de perturbação da norma L_∞ diferentes (capítulo 5.2). E por fim, o foco incide nos resultados de verificação formal aplicada a cada uma das 4 imagens de teste e a cada propriedade de robustez estipulada previamente (capítulo 5.3). No final de cada subcapítulo, encontra-se uma discussão no que diz respeito aos resultados alcançados.

Todos os ensaios experimentais foram implementados no software MATLAB, mais precisamente na versão MATLAB 2023b, com a ferramenta complementar NNV para realizar o método de verificação formal. Os resultados obtidos não recorreram ao uso de GPUs e foram realizados num computador pessoal com um CPU Intel(R) Core(TM) i5-7300HQ, onde a compilação e os cálculo de verificação formal foram executados.

5.1 TREINO E VALIDAÇÃO DAS REDES NEURONAIS UTILIZADAS NA CLASSIFICAÇÃO DE SINAIS DE TRÂNSITO

Após a construção dos 3 modelos de redes neuronais convolucionais, foi iniciado o processo de treino e validação de cada modelo, através da especificação de diferentes hiperparâmetros, de modo a encontrar pequenas características que pudessem favorecer o desempenho dos mesmos. O MATLAB oferece muitas opções de especificação para diferentes hiperparâmetros, mas para o caso deste trabalho, como se pode observar através da Tabela 5.1, foram utilizados e definidos a taxa de aprendizagem, o número máximo de épocas que serão aplicadas para treinar o modelo, o tamanho de lote de uma iteração, a taxa de divisão do conjunto de dados de treino em dados de treino e de validação, a frequência de validação, o algoritmo de otimização e o *verbose*.

A taxa de aprendizagem, que é o primeiro hiperparâmetro indicado na Tabela 5.1, descreve a taxa de atualização dos pesos em cada neurónio de uma rede neuronal, e é normalmente definida entre 0 e 1, sendo que neste caso o seu valor é 0.01. Ou seja, quanto maior for o valor da taxa de aprendizagem, maior é a taxa de atualização dos pesos, mas é necessário ter em conta o tempo de computação e também o facto de que para um valor muito baixo, os pesos dos neurónios praticamente não se alteram,

Parâmetro	Valor
Taxa de Aprendizagem	0.01
Máximo de Épocas	20
Tamanho de Lote (Treino)	128 imagens
Tamanho de Lote (Validação)	128 imagens
Taxa de Divisão	70/30
Frequência de Validação	30 iterações
Otimizador	SGD
Verbose	1

Tabela 5.1: Hiperparâmetros para o treino e validação das três redes neuronais desenvolvidas.

e para um valor muito alto, os pesos dos neurónios já podem ter sido aplicados a várias alterações. Como o valor de 0.01 é um muito utilizado na comunidade de investigação no domínio do DL e ML para o treino de modelos de redes neuronais de classificação de imagens, então será esse o utilizado neste trabalho.

O hiperparâmetro seguinte, isto é, o número máximo de épocas de treino, indica quantas vezes a rede neuronal deve percorrer um processo de treino completo, ou seja, neste caso, o modelo vai analisar as 39209 imagens do conjunto de dados de treino, bem como realizar a sua validação com 30% dessas imagens durante, no máximo, 20 vezes.

O tamanho de lote é um hiperparâmetro importante, responsável por definir a quantidade de amostras/imagens do conjunto de dados de treino que são utilizadas em cada iteração de cada época de treino (uma época precisa de percorrer todas as imagens/amostras de treino). Este parâmetro tem impacto tanto no desempenho como na velocidade de treino, porque é o parâmetro que dita o número de iterações para cada época de treino. O número de iterações em cada época é igual ao tamanho total do conjunto de dados de treino sobre o tamanho de cada lote, e no caso de se definir um valor muito grande para o tamanho de cada lote de treino, a qualidade do modelo de rede neuronal pode mesmo degradar-se porque o processo de treino não foi bem executado. Dessa forma, utilizou-se mini-lotes definindo um baixo valor de 128, o que reduz assim os requisitos de capacidade computacional durante o treino e ajuda na generalização do processo.

A frequência de validação e a taxa de divisão dos dados também são hiperparâmetros muito importantes, porque são os responsáveis de determinar se está a ocorrer *overfitting* dos dados de treino no modelo, ou seja, a validação em intervalos regulares durante o treino, para o caso deste trabalho de 30 iterações, ajuda a determinar se a rede está a se ajustar excessivamente aos dados de treino. Caso as métricas de treino (precisão e valor de perda) forem significativamente superiores que as métricas de validação, então a rede pode estar a obter o fenómeno de *overfitting*.

Relativamente aos dois últimos hiperparâmetros, o algoritmo de otimização executado é o SGD porque é o otimizador mais eficaz para o procedimento de treino do modelo com uma potência computacional mais baixa (sem ser topo de gama), e definir a opção *verbose* como 1 significa que o progresso das métricas de treino do modelo que está a ser treinado será amostrado ao longo do tempo de execução.

5.1.1 *Resultados*

Nesta subsecção, são apresentados os resultados das curvas obtidas da variação da precisão e do valor de perda a partir dos três modelos de CNN propostos (Figuras

5.1, 5.2 e 5.3), e também as três figuras de validação do treino realizado onde se efetuou a classificação de 6 imagens sem nenhum ataque do conjunto de dados de teste da base de dados GTSRB a cada um dos 3 modelos treinados (Figuras 5.4, 5.5 e 5.6). Na Tabela 5.2 realiza-se a comparação dos melhores resultados alcançados no processo de treino e validação para cada modelo, ou seja, o melhor valor de precisão e de perda mínima, o valor do número de parâmetros e o tempo necessário para a conclusão do treino. Na Tabela 5.3 efetua-se a comparação dos resultados dos três modelos criados neste trabalho com os diferentes modelos de redes neuronais mencionados no estado da arte para classificação de imagens de sinais de trânsito, mais precisamente, os modelos referenciados no capítulo 3.2.

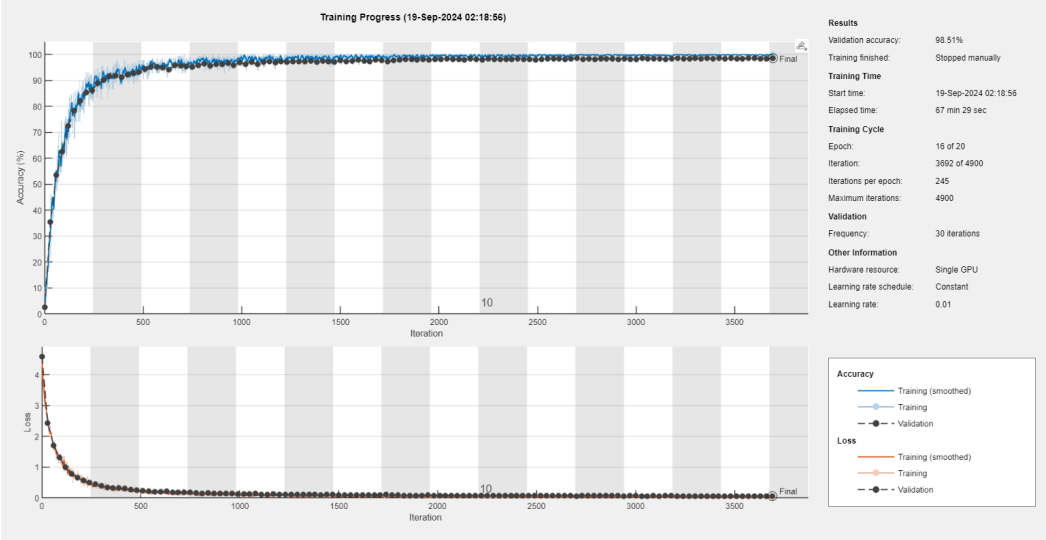

Figura 5.1: Curvas de precisão e de perda do processo de treino do modelo CNN 1.

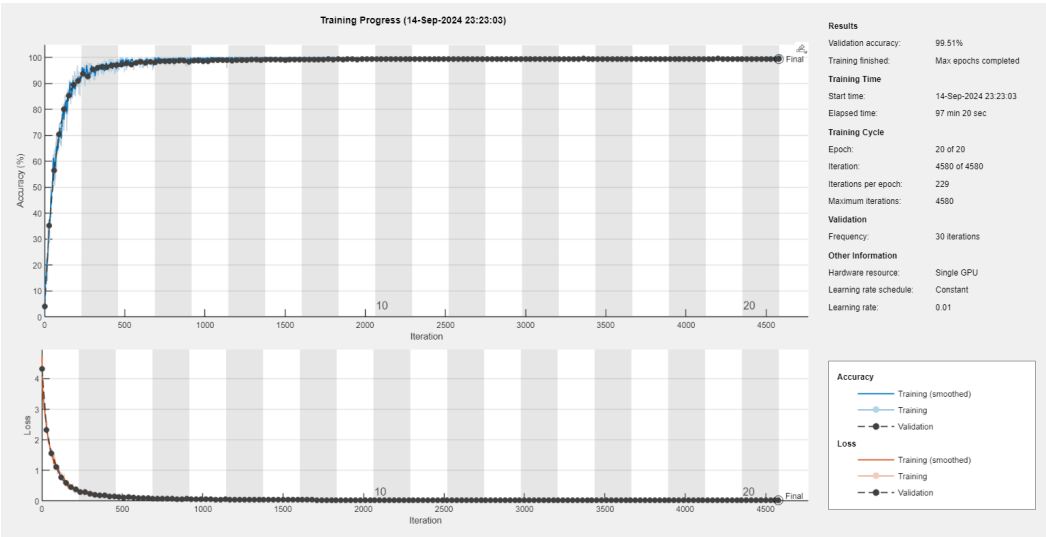

Figura 5.2: Curvas de precisão e de perda do processo de treino do modelo CNN 2.

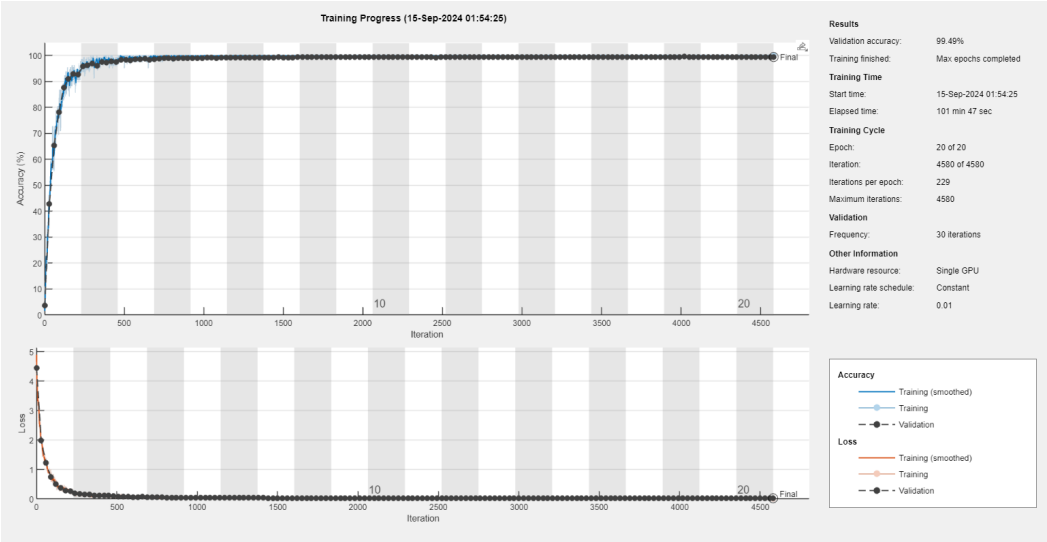

Figura 5.3: Curvas de precisão e de perda do processo de treino do modelo CNN 3.

Figura 5.4: Teste de validação do processo de treino do modelo CNN 1 com 6 imagens aleatórias do conjunto de dados de teste GTSRB.

Figura 5.5: Teste de validação do processo de treino do modelo CNN 2 com 6 imagens aleatórias do conjunto de dados de teste GTSRB.

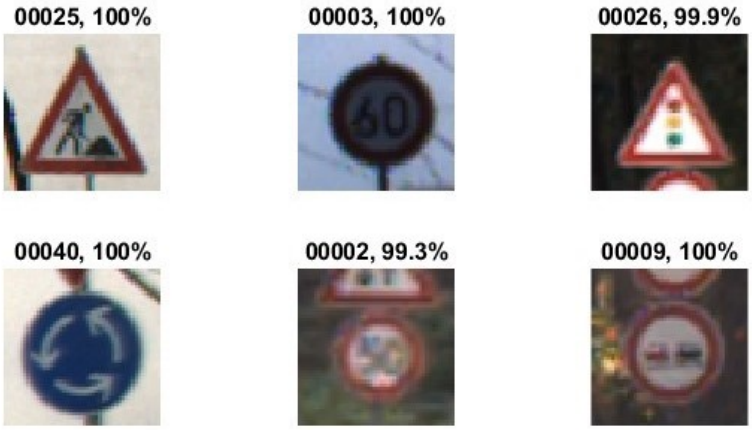

Figura 5.6: Teste de validação do processo de treino do modelo CNN 3 com 6 imagens aleatórias do conjunto de dados de teste GTSRB.

Modelos	Precisão (%)		Valor de perda	Número de Parâmetros (k)	Tempo de execução (h:min:s)
	Validação	Teste			
CNN 1	98.51	90.88	0.0657	45.5	01:06:48
CNN 2	99.51	95.13	0.0229	67.9	01:36:52
CNN 3	99.49	95.25	0.0232	122.9	01:41:19

Tabela 5.2: Resultados detalhados sobre o processo de treino e validação dos 3 modelos propostos.

Artigo (Autores e ano de publicação)	Precisão (%)	Nº de Parâmetros (k)
CNN 3 proposta	95.25	122.9
Cireşan et al. (2011) [49]	99.46	-
Kumar et al. (2022) [52]	98.81	240
Ferencz et al. (2024) [53]	97.98	≈2700

Tabela 5.3: Comparação da performance do melhor resultado de treino de um modelo de rede neuronal proposto com 3 trabalhos citados no estado da arte que utilizam a base de dados GTSRB.

5.1.2 *Discussão*

Nas Figuras 5.1, 5.2 e 5.3 estão representados os resultados das curvas obtidas a partir dos três modelos de CNN criados com, principalmente, diferentes formatos de imagem de entrada, isto é, 30x30x3, 32x32x3 e 48x48x3, ao longo de 20 épocas de treino, sendo que para o caso do modelo CNN 1 e diferenciando-se dos outros dois, o treino foi parado manualmente ao fim de 15 épocas de treino porque a variação dos valores de precisão e de perda entre sucessivas iterações já não era significativa. A

partir das três figuras, tanto para a curva de precisão, como para a curva dos valores de perda, pode-se constatar que, durante o treino, ambos os 3 modelos possuem uma boa inicialização e um bom traço sem oscilações, o que significa que se evitou o fenómeno do *overfitting* para todos os modelos CNN 1, CNN 2 e CNN 3.

Para as Figuras 5.4, 5.5 e 5.6, como modo de validação das curvas obtidas no processo de treino, é possível indicar que todos os 18 exemplos de classificação de imagens de sinais de trânsito presentes no conjunto de dados de teste da base de dados GTSRB foram bem classificados, e praticamente todas as percentagens de confiança de classificação encontram-se próximas dos 100%.

Relativamente à Tabela 5.2, os resultados relativos à precisão de validação e de teste são essencialmente os esperados, já que como o modelo CNN 3 possui o maior número de parâmetros, então irá ter uma maior precisão de classificação que todos os outros modelos. Como ambas as precisões de validação e de teste para todos os modelos testados não diferem consideravelmente entre si, então é possível concluir, mais uma vez, que todos os modelos não possuem a ocorrência de *overfitting*. Em termos do valor de perda, tal como o esperado, o valor diminui com o aumento da profundidade e complexidade da rede neuronal e também com o aumento de épocas de treino. E, em termos de número de parâmetros e tempo de execução do processo de treino, tal como era expectável, o número de parâmetros e o tempo de concretização do processo de treino aumenta com o aumento do número de camadas e do formato de imagem de entrada da uma rede neuronal. O melhor resultado encontrado é então, o do modelo CNN 3 com 95.25% de precisão e apenas 122923 parâmetros. A partir da Tabela 5.3, é também possível observar que o melhor resultado deste trabalho não ultrapassa nenhum dos artigos de diferentes classificadores de imagens de sinais de trânsito para o mesmo conjunto de dados de teste da base de dados GTSRB, citados no estado da arte, desenvolvidos ao longo dos anos, sendo que esse resultado era expectável de acontecer, já que é o modelo que contém o menor número de parâmetros de rede neuronal utilizados entre todos.

5.2 APLICAÇÃO DE ATAQUES ADVERSARIAIS

Apesar de os resultados da Tabela 5.2 indicarem que o modelo desenvolvido que possui uma maior percentagem de precisão de classificação e menor valor de perda é o CNN 3, como a diferença dos valores de precisão e de valor de perda entre esse modelo e o modelo de menor número de parâmetros, ou seja, o CNN 1, não é excessiva, então o único modelo de CNN concebido que será testado para a aplicação de ataques adversariais e para progressivamente a verificação formal, é o modelo CNN 1. Esta decisão baseia-se no facto de como o módulo computacional onde os resultados serão calculados não compreende grande capacidade computacional,

então, de modo a otimizar o tempo de cálculo de cada resultado experimental, apenas foram realizadas experiências utilizando o modelo CNN 1.

Os dois ataques realizados, tal como já mencionado anteriormente, são o FGSM e o PGD. Ambas as perturbações dos dois ataques são produzidas através do cálculo do gradiente de forma a obter o valor máximo de perda e a maior hipótese de uma classificação errada, mas enquanto o ataque FGSM só necessita de uma iteração, o ataque PGD procura o gradiente que produz um valor de perda superior através de diversas iterações. Deste modo, como ambos os ataques realizados neste trabalho utilizam a norma de perturbação L_∞, então para o ataque FGSM só é necessário especificar o valor limite de perturbação (ϵ), e para o ataque PGD é necessário especificar o número limite de iterações (*numIterações*), o passo para o cálculo do gradiente entre cada iteração (α) e o valor limite de perturbação (ϵ). No total, de forma a verificar as propriedades de robustez estipuladas, foram realizados 5 ensaios de perturbações diferentes com o valor limite de perturbação para cada ataque igual aos valores estipulados nas propriedades de robustez. O conjunto das especificações de cada ensaio estão indicadas abaixo seguindo o modelo $especificaçãoParâmetro = \{\text{ensaio 1; ensaio 2; ensaio 3; ensaio 4; ensaio 5}\}$:

- **Ataque FGSM:** $\epsilon = \{0.1; 0.5; 1; 2; 3\}$.
- **Ataque PGD:** $numIterações = \{50; 50; 50; 50; 50\}$, $\alpha = \{0.01; 0.05; 0.1; 0.2; 0.3\}$ e $\epsilon = \{0.1; 0.5; 1; 2; 3\}$.

Relativamente às 4 imagens do conjunto de dados de teste da base de dados GTSRB ao qual foram aplicados os dois ataques, inicialmente foi necessário realizar o seu pré-processamento para o formato de dados de entrada do modelo CNN 1, ou seja, 30x30x3, e posteriormente é que foram produzidos os exemplos adversariais para os diferentes valores de perturbação da norma L_∞. Nas duas figuras seguintes estão ilustrados todos os sinais classificados pelo modelo CNN 1 para todos os 5 ensaios realizados de aplicação do ataque FGSM e PGD. Na Figura 5.7 estão representados os 4 sinais de trânsito bem classificados, com a classe respetiva, e na Figura 5.8 estão representados os 3 sinais de trânsito, com o *label* respetivo, nos quais o modelo CNN 1 classificou incorretamente. Isto é, ambos os ataques conseguiram pelo menos enganar o modelo a classificar erradamente um sinal de trânsito de 3 das 4 imagens retiradas do conjunto de dados de teste.

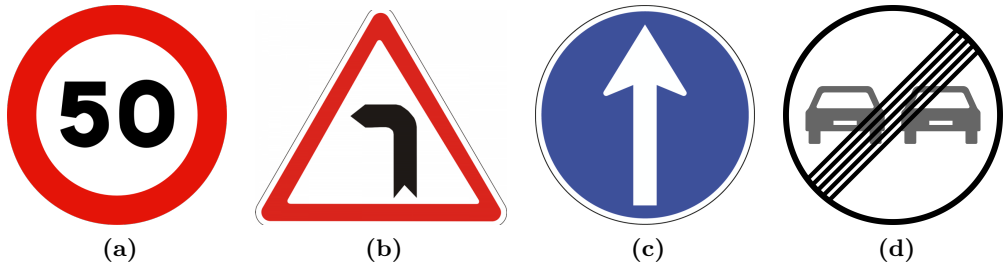

Figura 5.7: Representação translúcida dos 4 sinais de trânsito das 4 imagens de teste que são aplicadas a 2 ataques adversariais: (a) Classe: 00002; (b) Classe: 00019; (c) Classe: 00035; (d) Classe: 00041.

Figura 5.8: Representação translúcida dos 3 sinais de trânsito mal classificados de 3 das 4 imagens de teste que são aplicadas a 2 ataques adversariais: (a) Classe: 00005; (b) Classe: 00023; (c) Classe: 00032.

5.2.1 *Resultados do ataque FGSM*

Nesta subsecção, são apresentados os resultados experimentais do ataque FGSM às 4 imagens de sinais de trânsito do conjunto de dados de teste representados de forma nítida na Figura 5.7. São obtidos o exemplo adversarial (imagem + perturbação), a perturbação e a classe (*label*) do sinal de trânsito classificado antes e depois do ataque por meio do modelo CNN 1. Nas Figuras 5.9, 5.14, 5.19 e 5.24 são apresentados os resultados para o ataque FGSM com valor de $\epsilon = 0.1$, nas Figuras 5.10, 5.15, 5.20 e 5.25 são apresentados os resultados para o ataque FGSM com valor de $\epsilon = 0.5$, nas Figuras 5.11, 5.16, 5.21 e 5.26 são apresentados os resultados para o ataque FGSM com valor de $\epsilon = 1$, nas Figuras 5.12, 5.17, 5.22 e 5.27 são apresentados os resultados para o ataque FGSM com valor de $\epsilon = 2$, e nas Figuras 5.13, 5.18, 5.23 e 5.28 são apresentados os resultados para o ataque FGSM com valor de $\epsilon = 3$.

Figura 5.9: Resultado de classificação da imagem do sinal de trânsito de rótulo 00002 com o ataque FGSM para um $\epsilon = 0.1$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

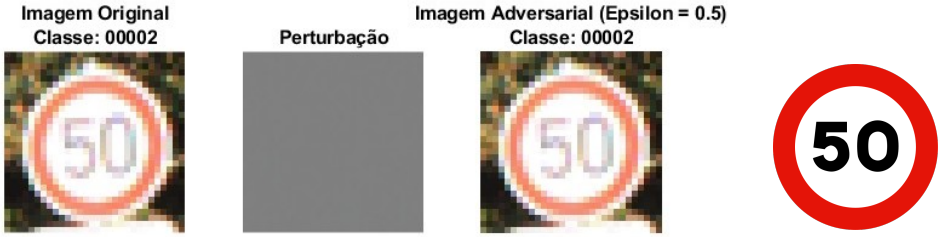

Figura 5.10: Resultado de classificação da imagem do sinal de trânsito de rótulo 00002 com o ataque FGSM para um $\epsilon = 0.5$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.11: Resultado de classificação da imagem do sinal de trânsito de rótulo 00002 com o ataque FGSM para um $\epsilon = 1$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.12: Resultado de classificação da imagem do sinal de trânsito de rótulo 00002 com o ataque FGSM para um $\epsilon = 2$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.13: Resultado de classificação da imagem do sinal de trânsito de rótulo 00002 com o ataque FGSM para um $\epsilon = 3$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.14: Resultado de classificação da imagem do sinal de trânsito de rótulo 00019 com o ataque FGSM para um $\epsilon = 0.1$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.15: Resultado de classificação da imagem do sinal de trânsito de rótulo 00019 com o ataque FGSM para um $\epsilon = 0.5$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.16: Resultado de classificação da imagem do sinal de trânsito de rótulo 00019 com o ataque FGSM para um $\epsilon = 1$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.17: Resultado de classificação da imagem do sinal de trânsito de rótulo 00019 com o ataque FGSM para um $\epsilon = 2$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.18: Resultado de classificação da imagem do sinal de trânsito de rótulo 00019 com o ataque FGSM para um $\epsilon = 3$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.19: Resultado de classificação da imagem do sinal de trânsito de rótulo 00035 com o ataque FGSM para um $\epsilon = 0.1$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.20: Resultado de classificação da imagem do sinal de trânsito de rótulo 00035 com o ataque FGSM para um $\epsilon = 0.5$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

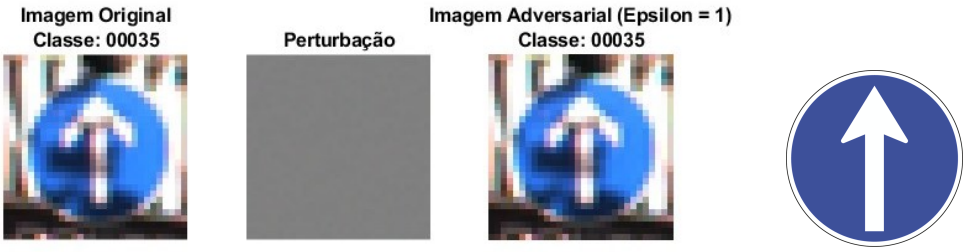

Figura 5.21: Resultado de classificação da imagem do sinal de trânsito de rótulo 00035 com o ataque FGSM para um $\epsilon = 1$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.22: Resultado de classificação da imagem do sinal de trânsito de rótulo 00035 com o ataque FGSM para um $\epsilon = 2$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.23: Resultado de classificação da imagem do sinal de trânsito de rótulo 00035 com o ataque FGSM para um $\epsilon = 3$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.24: Resultado de classificação da imagem do sinal de trânsito de rótulo 00041 com o ataque FGSM para um $\epsilon = 0.1$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.25: Resultado de classificação da imagem do sinal de trânsito de rótulo 00041 com o ataque FGSM para um $\epsilon = 0.5$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

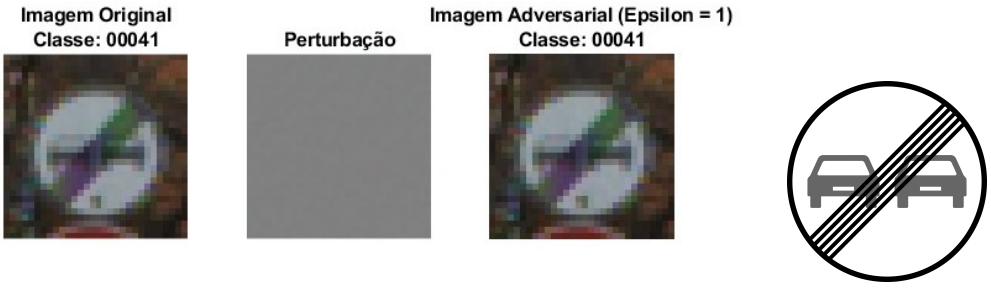

Figura 5.26: Resultado de classificação da imagem do sinal de trânsito de rótulo 00041 com o ataque FGSM para um $\epsilon = 1$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

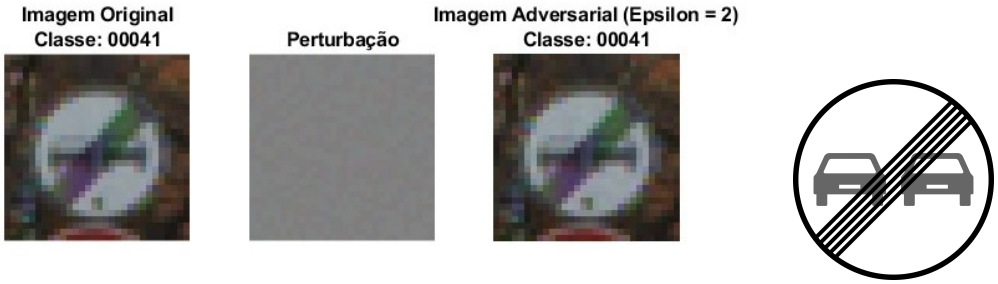

Figura 5.27: Resultado de classificação da imagem do sinal de trânsito de rótulo 00041 com o ataque FGSM para um $\epsilon = 2$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.28: Resultado de classificação da imagem do sinal de trânsito de rótulo 00041 com o ataque FGSM para um $\epsilon = 3$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

5.2.2 *Resultados do ataque PGD*

Nesta subsecção, de forma equivalente ao capítulo 5.2.1, são apresentados os resultados experimentais, mas desta vez, do ataque PGD às 4 imagens de teste. Nas Figuras 5.29, 5.34, 5.39 e 5.44 são apresentados os resultados para o ataque PGD com valor de $\epsilon = 0.1$, nas Figuras 5.30, 5.35, 5.40 e 5.45 são apresentados os resultados para o ataque PGD com valor de $\epsilon = 0.5$, nas Figuras 5.31, 5.36, 5.41 e 5.46 são apresentados os resultados para o ataque PGD com valor de $\epsilon = 1$, nas Figuras 5.32, 5.37, 5.42 e 5.47 são apresentados os resultados para o ataque PGD com valor de $\epsilon = 2$, e nas Figuras 5.33, 5.38, 5.43 e 5.48 são apresentados os resultados para o ataque PGD com valor de $\epsilon = 3$.

Figura 5.29: Resultado de classificação da imagem do sinal de trânsito de rótulo 00002 com o ataque PGD para um $\epsilon = 0.1$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.30: Resultado de classificação da imagem do sinal de trânsito de rótulo 00002 com o ataque PGD para um $\epsilon = 0.5$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.31: Resultado de classificação da imagem do sinal de trânsito de rótulo 00002 com o ataque PGD para um $\epsilon = 1$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.32: Resultado de classificação da imagem do sinal de trânsito de rótulo 00002 com o ataque PGD para um $\epsilon = 2$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.33: Resultado de classificação da imagem do sinal de trânsito de rótulo 00002 com o ataque PGD para um $\epsilon = 3$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.34: Resultado de classificação da imagem do sinal de trânsito de rótulo 00019 com o ataque PGD para um $\epsilon = 0.1$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.35: Resultado de classificação da imagem do sinal de trânsito de rótulo 00019 com o ataque PGD para um $\epsilon = 0.5$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.36: Resultado de classificação da imagem do sinal de trânsito de rótulo 00019 com o ataque PGD para um $\epsilon = 1$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.37: Resultado de classificação da imagem do sinal de trânsito de rótulo 00019 com o ataque PGD para um $\epsilon = 2$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.38: Resultado de classificação da imagem do sinal de trânsito de rótulo 00019 com o ataque PGD para um $\epsilon = 3$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.39: Resultado de classificação da imagem do sinal de trânsito de rótulo 00035 com o ataque PGD para um $\epsilon = 0.1$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.40: Resultado de classificação da imagem do sinal de trânsito de rótulo 00035 com o ataque PGD para um $\epsilon = 0.5$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.41: Resultado de classificação da imagem do sinal de trânsito de rótulo 00035 com o ataque PGD para um $\epsilon = 1$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

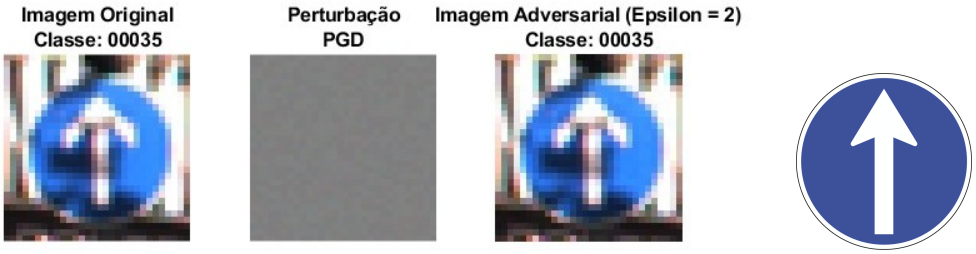

Figura 5.42: Resultado de classificação da imagem do sinal de trânsito de rótulo 00035 com o ataque PGD para um $\epsilon = 2$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.43: Resultado de classificação da imagem do sinal de trânsito de rótulo 00035 com o ataque PGD para um $\epsilon = 3$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

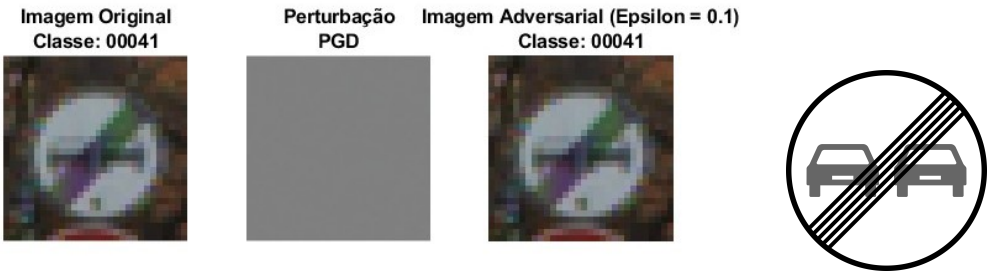

Figura 5.44: Resultado de classificação da imagem do sinal de trânsito de rótulo 00041 com o ataque PGD para um $\epsilon = 0.1$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.45: Resultado de classificação da imagem do sinal de trânsito de rótulo 00041 com o ataque PGD para um $\epsilon = 0.5$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.46: Resultado de classificação da imagem do sinal de trânsito de rótulo 00041 com o ataque PGD para um $\epsilon = 1$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.47: Resultado de classificação da imagem do sinal de trânsito de rótulo 00041 com o ataque PGD para um $\epsilon = 2$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

Figura 5.48: Resultado de classificação da imagem do sinal de trânsito de rótulo 00041 com o ataque PGD para um $\epsilon = 3$. À direita da imagem adversarial encontra-se a representação translúcida do sinal de trânsito classificado.

5.2.3 *Discussão*

A partir de todas as 40 figuras apresentadas de aplicação dos ataques FGSM e PGD a 4 imagens do conjunto de dados de teste, é possível concluir que o modelo CNN 1 com o treino de aprendizagem que foi realizado ao longo de 15 épocas, só é plenamente robusto em todo o conjunto de valores de perturbação utilizados para o caso da imagem do sinal de obrigação de sentido sempre em frente, ou seja, do sinal com *label* 35. Enquanto o sinal de trânsito de classe 2 é robusto em todo o conjunto de valores limite de perturbação do ataque FGSM, para o caso do ataque PGD isso já não acontece, porque com um $\epsilon = 3$, o sinal de proibição de exceder a velocidade máxima de 50 km/h é mal classificado como um sinal de proibição de exceder a velocidade máxima de 80 km/h (classe 5). Para o caso dos sinais de trânsito de classe 19 e 41, nem são robustos no conjunto de valores limite de perturbação para o ataque FGSM e para o ataque PGD, visto que o sinal de perigo com curva à esquerda (*label* 19) com um $\epsilon = 3$ para o ataque FGSM e para um conjunto de $\epsilon = \{2; 3\}$ para o ataque PGD é classificado incorretamente como um sinal de perigo de pavimento escorregadio (classe 23), e o sinal de fim de proibição de ultrapassar (classe 41) para um mesmo valor de ϵ para o ataque FGSM e conjunto de valores de

ϵ para o ataque PGD é classificado erradamente como um sinal de fim de todas as proibições anteriores (classe 32).

Estas soluções vão ao encontro dos resultados obtidos pelo autor que desenvolveu o ataque PGD [30], onde se demonstra que as perturbações aplicadas e geradas pelo ataque *Projected Gradient Descent* (PGD) são mais fortes e intensas que as perturbações criadas pelo ataque FGSM.

5.3 VERIFICAÇÃO FORMAL

A ferramenta de verificação formal utilizada é a NNV, e o método de verificação usado nesta ferramenta é o *ImageStar*. O método *ImageStar* aborda o problema de verificação formal de redes neuronais a partir de um problema de alcançabilidade, e por isso, tanto pode possuir uma abordagem de alcançabilidade sobre-aproximada, ou uma abordagem de alcançabilidade exata. Como citado no capítulo 4.4.2, caso seja utilizada uma abordagem de alcançabilidade exata, o verificador do NNV ou retorna o resultado de "robusto", ou "não robusto", juntamente com um conjunto de contra-exemplos, e caso seja utilizada uma abordagem de alcançabilidade sobre-aproximada, o verificador do NNV ou devolve o resultado de "robusto", ou "resultado desconhecido", uma vez que a ferramenta não consegue comprovar a não robustez do modelo a ser verificado, já que está a efetuar uma *"aproximação"* sobre-aproximada dos intervalos de alcançabilidade de saída. De modo a se obter um procedimento de verificação formal mais atingível e eficaz em relação ao intervalo de computação esperado, tendo em conta o módulo computacional usado, foram obtidos resultados única e exclusivamente para o método de alcançabilidade *ImageStar* com uma abordagem sobre-aproximada.

Tal como nos resultados de aplicação de ataques adversariais, o único modelo de rede neuronal empregue do processo de verificação formal é o CNN 1, e o conjunto de valores de entrada usado na verificação também são as 4 imagens que foram sobrepostas aos dois ataques adversariais.

A ferramenta NNV, apesar de indicar no seu documento técnico [56] que aplica um ataque adversarial de norma de perturbação L_∞ ao conjunto de valores de entrada, o que acontece na verdade é que para uma determinada propriedade de robustez com um valor limite (ϵ) de perturbação estipulado e para uma determinada imagem de um sinal de trânsito de entrada, o conjunto de valores de entrada é criado alterando todos os valores de pixeis da imagem para todos os canais RGB entre um intervalo de $[-\epsilon, +\epsilon]$, ou seja, todos os valores de pixeis da imagem de entrada são reduzidos e aumentados ϵ unidades. Isso leva à criação de uma imagem mais escurecida e outra imagem mais clara relativamente à imagem de entrada, o

que representa, respetivamente, o limite inferior e superior do conjunto ou intervalo de valores de entrada. Na Figura 5.49 estão representados os dois limites do conjunto de entrada de uma das imagens usadas do conjunto de teste para o método de alcançabilidade *ImageStar*, com um valor limite de perturbação (ϵ) igual a 26, isto é, todos os valores de pixeis da imagem foram submetidos a uma diminuição e aumento de 26 unidades.

Figura 5.49: Ilustração de uma imagem de sinal de trânsito para a aplicação do método *ImageStar* com a criação do conjunto de valores de entrada entre uma imagem mais escurecida (*lower bound*) e uma imagem mais clara (*upper bound*).

Depois da criação do conjunto de valores de entrada pela ferramenta NNV, é realizado o cálculo dos conjuntos de valores de alcançabilidade da última camada do modelo utilizado antes da camada *Softmax*, para todas as 43 classes da base de dados GTSRB. E, os valores de alcançabilidade obtidos dependem fortemente do tipo de rede neuronal utilizada, porque caso seja utilizada uma rede neuronal profunda de múltiplas camadas convolucionais e totalmente conectadas, então o valor de saída de cada intervalo de alcançabilidade vai aumentar, uma vez que com um maior número de camadas, então irão ser retiradas mais características do sinal de trânsito da imagem de entrada, e isso leva a um maior tempo de cálculo.

Na hipótese de algum intervalo de alcançabilidade final intersetar entre duas classes de sinais de trânsito da base de dados GTSRB para o conjunto de valores de entrada durante a análise da alcançabilidade do modelo, então o modelo para esse conjunto de valores de entrada e para essa especificação de propriedade de robustez, não é robusto para o método de alcançabilidade *ImageStar* exato e apresenta um resultado desconhecido para o método de alcançabilidade *ImageStar* sobre-aproximado.

Por outras palavras, o método de verificação formal utilizado como um problema de alcançabilidade neste trabalho pode ser dividido em 3 etapas. Primeiramente, a imagem adversarial em teste com a perturbação aplicada é representada num intervalo de alcançabilidade entre um limite inferior e superior. Seguidamente, esse intervalo/conjunto de alcançabilidade da imagem adversarial vai ser passado pelo modelo de rede neuronal, e irão ser calculados e refinados camada a camada através

de todas as operações em cada camada da rede, os conjuntos alcançáveis de saída para todas as saídas possíveis do modelo, ou seja, irão ser calculados todos os 43 intervalos/conjuntos de alcançabilidade de saída correspondentes a cada uma das 43 classes de sinais de trânsito da base de dados GTSRB. No final, é gerado um gráfico com os intervalos/conjuntos de alcançabilidade de todas as saídas possíveis do modelo, e caso algum intervalo/conjunto de alcançabilidade de saída sem ser a classe verdadeira, interseta com o conjunto/intervalo de alcançabilidade de saída correspondente à classe verdadeira do sinal de trânsito em teste, então o modelo de rede neuronal, supostamente, não é robusto para a propriedade de robustez (e valor limite de perturbação) especificada.

5.3.1 *Resultados*

Nesta subsecção, são apresentados os resultados da verificação de robustez e da análise de alcançabilidade do modelo CNN 1 para as 4 imagens de teste utilizadas e 5 propriedades de robustez estipuladas. Na Tabela 5.4 realiza-se a comparação dos resultados alcançados no processo de verificação formal da robustez do modelo CNN 1, onde se indica a imagem de teste e a propriedade utilizadas, o resultado da verificação, e o tempo que foi necessário para a sua conclusão. E nas Figuras 5.50 até 5.67, são evidenciados os gráficos de comparação dos conjuntos alcançáveis para cada classe de cada 1 dos 20 ensaios experimentais executados, caso o processo de verificação não ultrapasse o intervalo de tempo estabelecido (6 horas ou 21600 segundos).

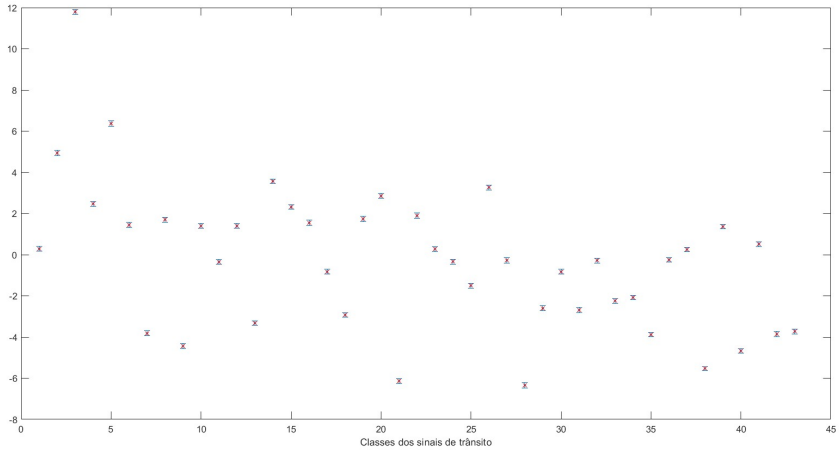

Figura 5.50: Gráfico de comparação dos conjuntos alcançáveis para cada classe a partir da imagem de entrada de classe 2 com um $\epsilon = 0.1$. O eixo das abcissas representa a classe do sinal de trânsito e o eixo das ordenadas representa os "logits" de alcançabilidade na saída da camada totalmente conectada que precede a camada de ativação Softmax.

Imagem de Entrada	Propriedade de Robustez	Resultado de Verificação	Tempo de Execução (s)
Classe: 2	ϕ_1 ($\epsilon = 0.1$) [-0.1,+0.1]	Verificado para ser robusto	7.58
	ϕ_2 ($\epsilon = 0.5$) [-0.5,+0.5]	Verificado para ser robusto	14.17
	ϕ_3 ($\epsilon = 1.0$) [-1.0,+1.0]	Verificado para ser robusto	33.67
	ϕ_4 ($\epsilon = 2.0$) [-2.0,+2.0]	Resultado desconhecido	1162.60
	ϕ_5 ($\epsilon = 3.0$) [-3.0,+3.0]	Resultado desconhecido	2261.71
Classe: 19	ϕ_1 ($\epsilon = 0.1$) [-0.1,+0.1]	Verificado para ser robusto	6.77
	ϕ_2 ($\epsilon = 0.5$) [-0.5,+0.5]	Verificado para ser robusto	47.36
	ϕ_3 ($\epsilon = 1.0$) [-1.0,+1.0]	Verificado para ser robusto	180.08
	ϕ_4 ($\epsilon = 2.0$) [-2.0,+2.0]	Resultado desconhecido	5421.45
	ϕ_5 ($\epsilon = 3.0$) [-3.0,+3.0]	*Timeout*	-
Classe: 35	ϕ_1 ($\epsilon = 0.1$) [-0.1,+0.1]	Verificado para ser robusto	6.30
	ϕ_2 ($\epsilon = 0.5$) [-0.5,+0.5]	Verificado para ser robusto	13.29
	ϕ_3 ($\epsilon = 1.0$) [-1.0,+1.0]	Verificado para ser robusto	32.73
	ϕ_4 ($\epsilon = 2.0$) [-2.0,+2.0]	Verificado para ser robusto	110.01
	ϕ_5 ($\epsilon = 3.0$) [-3.0,+3.0]	Verificado para ser robusto	631.32
Classe: 41	ϕ_1 ($\epsilon = 0.1$) [-0.1,+0.1]	Verificado para ser robusto	8.65
	ϕ_2 ($\epsilon = 0.5$) [-0.5,+0.5]	Verificado para ser robusto	47.15
	ϕ_3 ($\epsilon = 1.0$) [-1.0,+1.0]	Verificado para ser robusto	173.85
	ϕ_4 ($\epsilon = 2.0$) [-2.0,+2.0]	Resultado desconhecido	4891.56
	ϕ_5 ($\epsilon = 3.0$) [-3.0,+3.0]	*Timeout*	-

Tabela 5.4: Resultados detalhados sobre o processo de verificação formal do modelo CNN 1 para as 4 imagens de teste e 5 propriedades de robustez definidas.

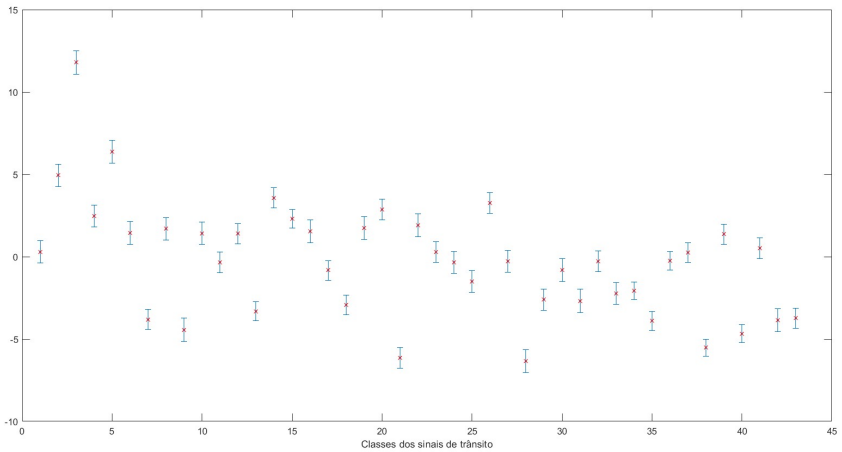

Figura 5.51: Gráfico de comparação dos conjuntos alcançáveis para cada classe a partir da imagem de entrada de classe 2 com um $\epsilon = 0.5$. O eixo das abcissas representa a classe do sinal de trânsito e o eixo das ordenadas representa os "logits" de alcançabilidade na saída da camada totalmente conectada que precede a camada de ativação Softmax.

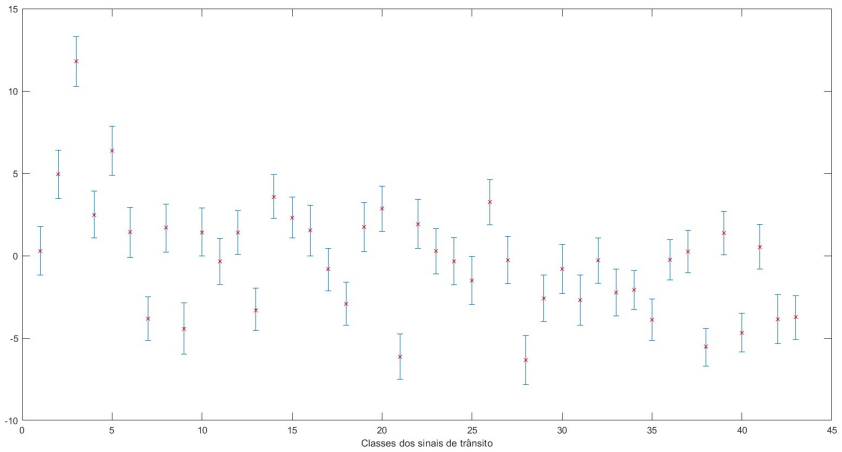

Figura 5.52: Gráfico de comparação dos conjuntos alcançáveis para cada classe a partir da imagem de entrada de classe 2 com um $\epsilon = 1$. O eixo das abcissas representa a classe do sinal de trânsito e o eixo das ordenadas representa os "logits" de alcançabilidade na saída da camada totalmente conectada que precede a camada de ativação Softmax.

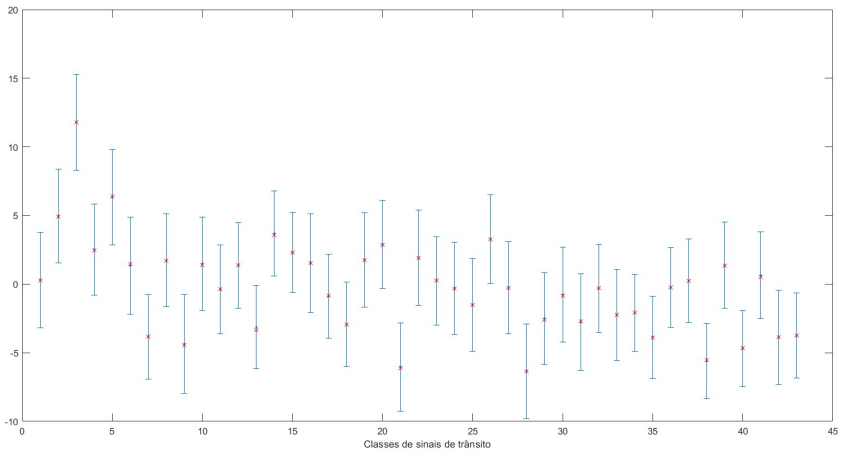

Figura 5.53: Gráfico de comparação dos conjuntos alcançáveis para cada classe a partir da imagem de entrada de classe 2 com um $\epsilon = 2$. O eixo das abcissas representa a classe do sinal de trânsito e o eixo das ordenadas representa os "logits" de alcançabilidade na saída da camada totalmente conectada que precede a camada de ativação Softmax.

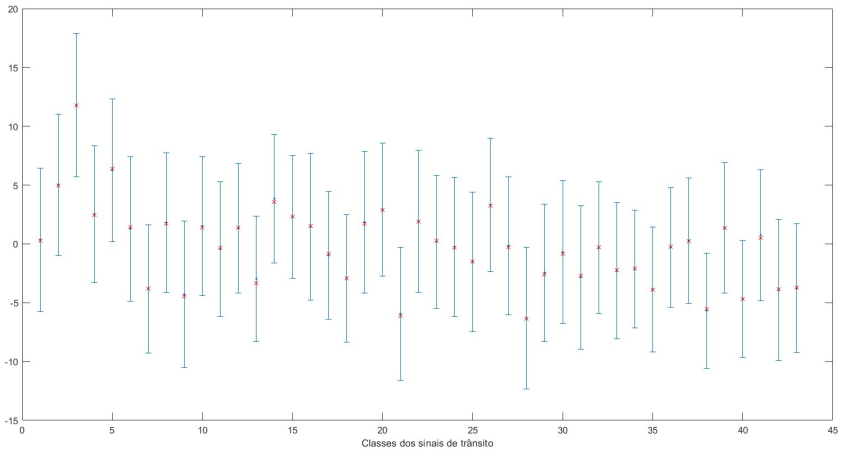

Figura 5.54: Gráfico de comparação dos conjuntos alcançáveis para cada classe a partir da imagem de entrada de classe 2 com um $\epsilon = 3$. O eixo das abcissas representa a classe do sinal de trânsito e o eixo das ordenadas representa os "logits" de alcançabilidade na saída da camada totalmente conectada que precede a camada de ativação Softmax.

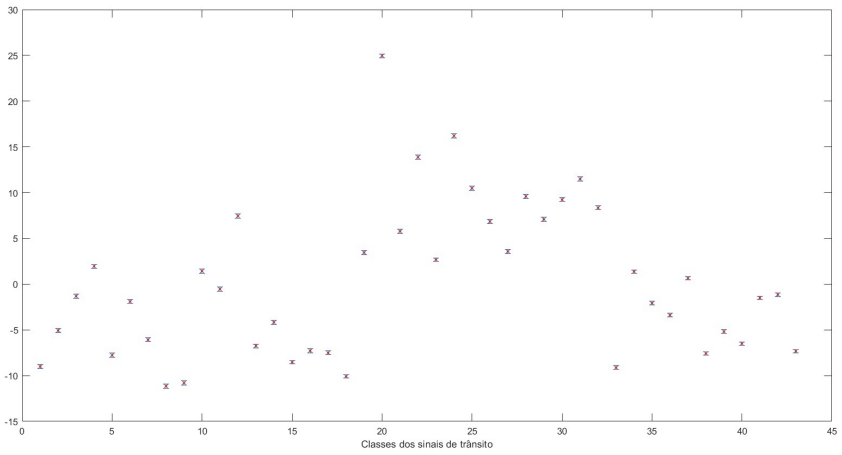

Figura 5.55: Gráfico de comparação dos conjuntos alcançáveis para cada classe a partir da imagem de entrada de classe 19 com um $\epsilon = 0.1$. O eixo das abcissas representa a classe do sinal de trânsito e o eixo das ordenadas representa os "logits" de alcançabilidade na saída da camada totalmente conectada que precede a camada de ativação Softmax.

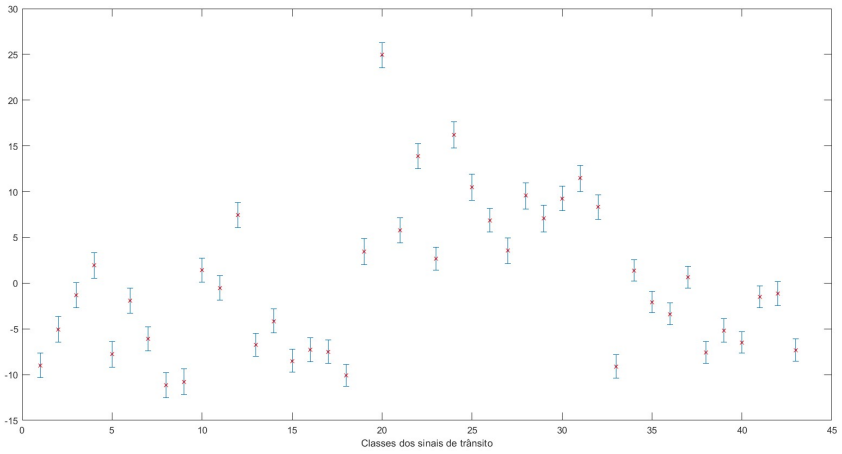

Figura 5.56: Gráfico de comparação dos conjuntos alcançáveis para cada classe a partir da imagem de entrada de classe 19 com um $\epsilon = 0.5$. O eixo das abcissas representa a classe do sinal de trânsito e o eixo das ordenadas representa os "logits" de alcançabilidade na saída da camada totalmente conectada que precede a camada de ativação Softmax.

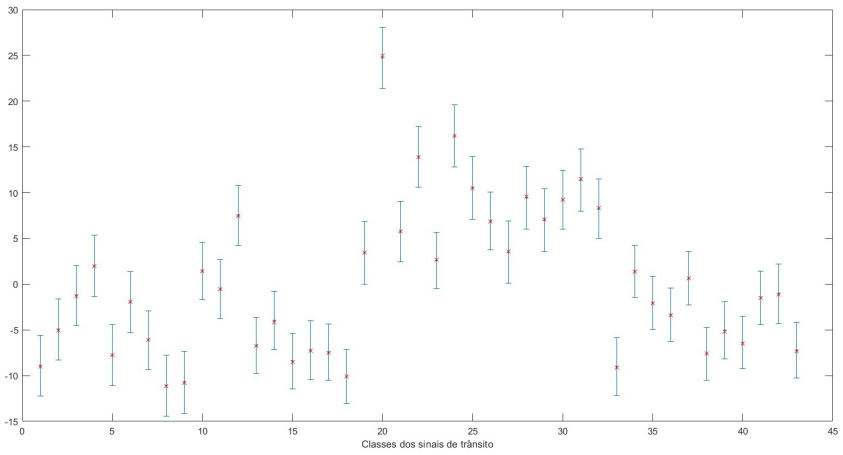

Figura 5.57: Gráfico de comparação dos conjuntos alcançáveis para cada classe a partir da imagem de entrada de classe 19 com um $\epsilon = 1$. O eixo das abcissas representa a classe do sinal de trânsito e o eixo das ordenadas representa os "logits" de alcançabilidade na saída da camada totalmente conectada que precede a camada de ativação Softmax.

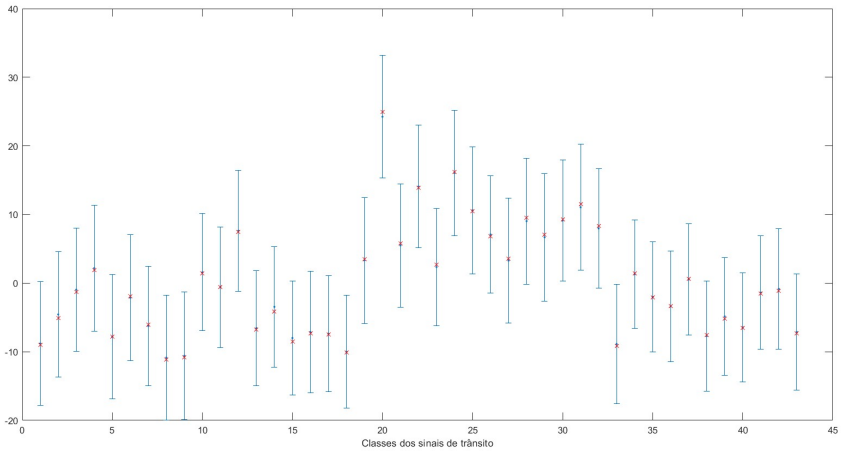

Figura 5.58: Gráfico de comparação dos conjuntos alcançáveis para cada classe a partir da imagem de entrada de classe 19 com um $\epsilon = 2$. O eixo das abcissas representa a classe do sinal de trânsito e o eixo das ordenadas representa os "logits" de alcançabilidade na saída da camada totalmente conectada que precede a camada de ativação Softmax.

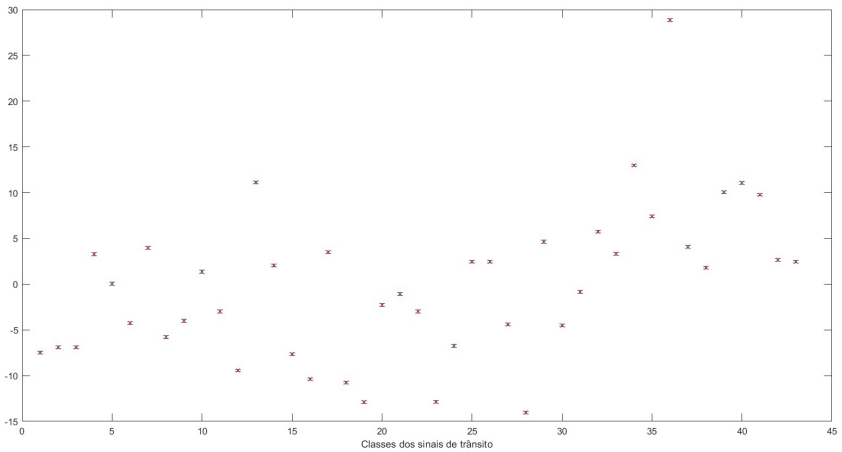

Figura 5.59: Gráfico de comparação dos conjuntos alcançáveis para cada classe a partir da imagem de entrada de classe 35 com um $\epsilon = 0.1$. O eixo das abcissas representa a classe do sinal de trânsito e o eixo das ordenadas representa os "logits" de alcançabilidade na saída da camada totalmente conectada que precede a camada de ativação Softmax.

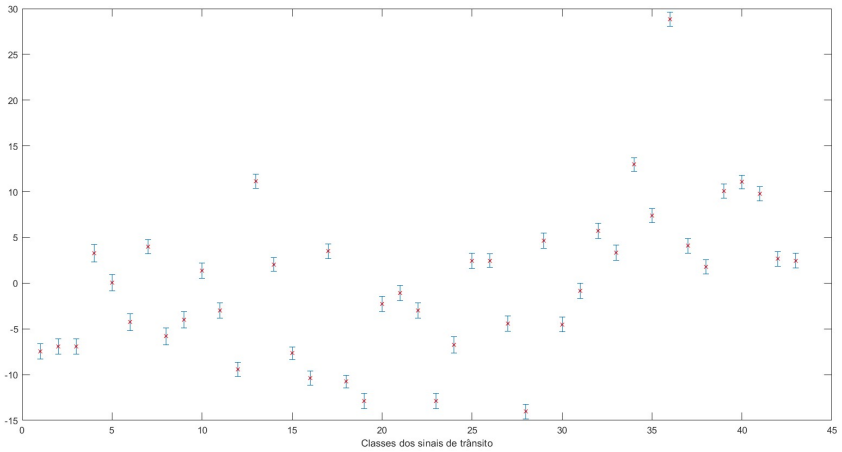

Figura 5.60: Gráfico de comparação dos conjuntos alcançáveis para cada classe a partir da imagem de entrada de classe 35 com um $\epsilon = 0.5$. O eixo das abcissas representa a classe do sinal de trânsito e o eixo das ordenadas representa os "logits" de alcançabilidade na saída da camada totalmente conectada que precede a camada de ativação Softmax.

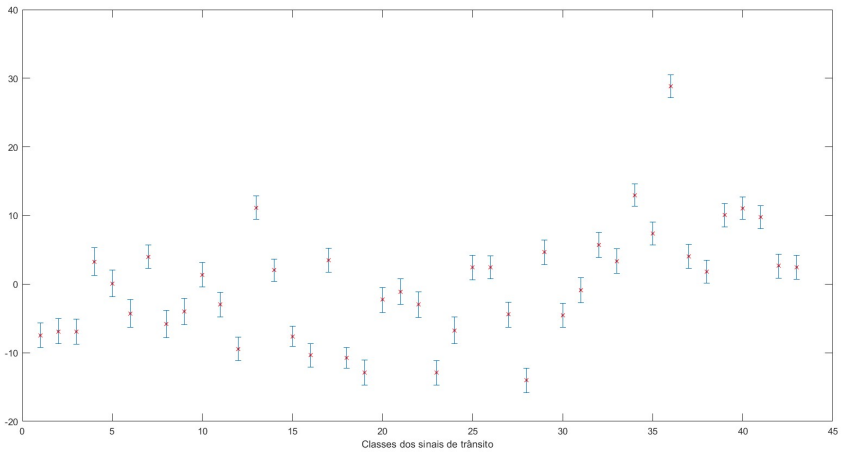

Figura 5.61: Gráfico de comparação dos conjuntos alcançáveis para cada classe a partir da imagem de entrada de classe 35 com um $\epsilon = 1$. O eixo das abcissas representa a classe do sinal de trânsito e o eixo das ordenadas representa os "logits" de alcançabilidade na saída da camada totalmente conectada que precede a camada de ativação Softmax.

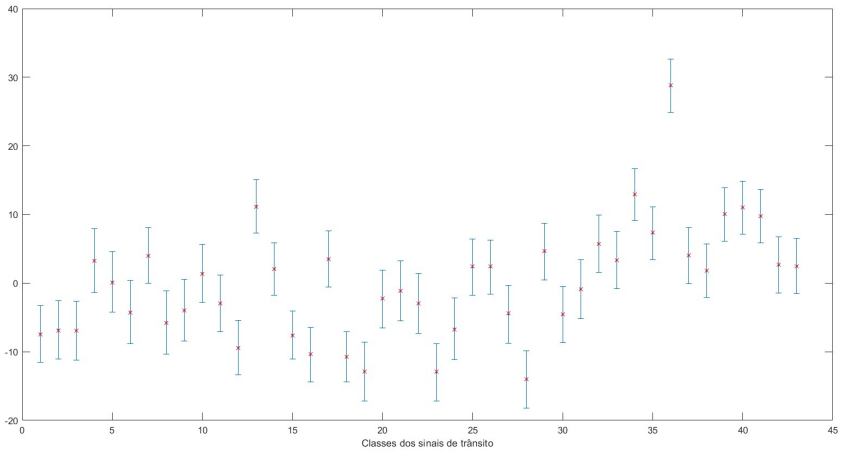

Figura 5.62: Gráfico de comparação dos conjuntos alcançáveis para cada classe a partir da imagem de entrada de classe 35 com um $\epsilon = 2$. O eixo das abcissas representa a classe do sinal de trânsito e o eixo das ordenadas representa os "logits" de alcançabilidade na saída da camada totalmente conectada que precede a camada de ativação Softmax.

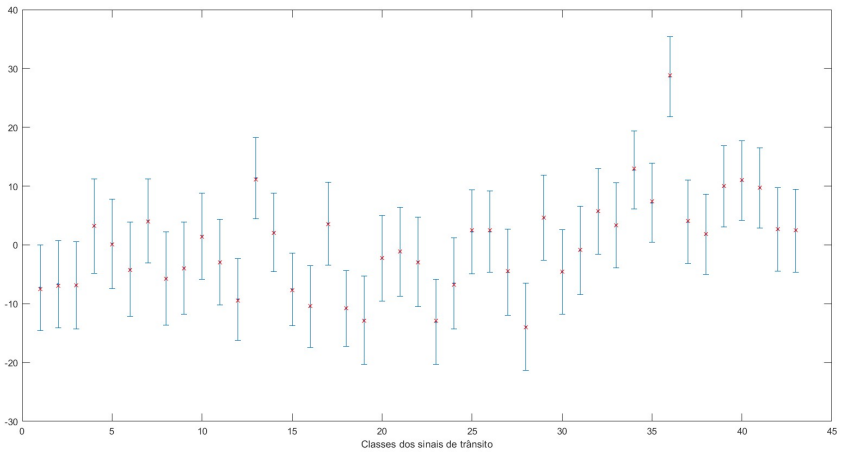

Figura 5.63: Gráfico de comparação dos conjuntos alcançáveis para cada classe a partir da imagem de entrada de classe 35 com um $\epsilon = 3$. O eixo das abcissas representa a classe do sinal de trânsito e o eixo das ordenadas representa os "logits" de alcançabilidade na saída da camada totalmente conectada que precede a camada de ativação Softmax.

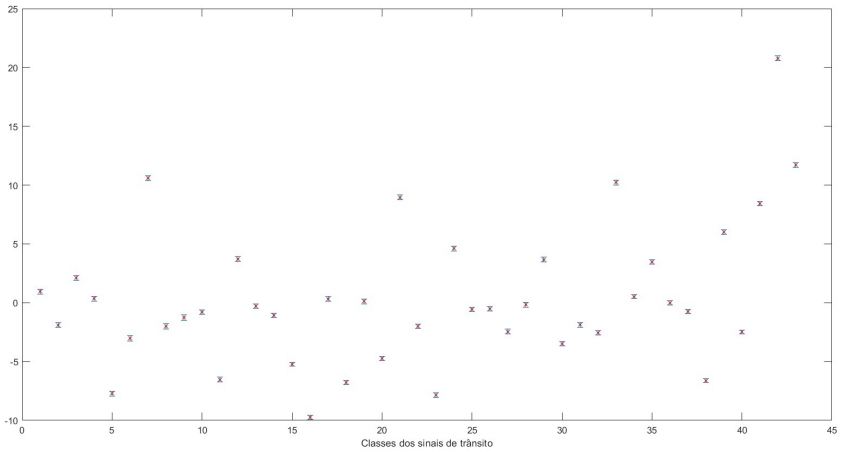

Figura 5.64: Gráfico de comparação dos conjuntos alcançáveis para cada classe a partir da imagem de entrada de classe 41 com um $\epsilon = 0.1$. O eixo das abcissas representa a classe do sinal de trânsito e o eixo das ordenadas representa os "logits" de alcançabilidade na saída da camada totalmente conectada que precede a camada de ativação Softmax.

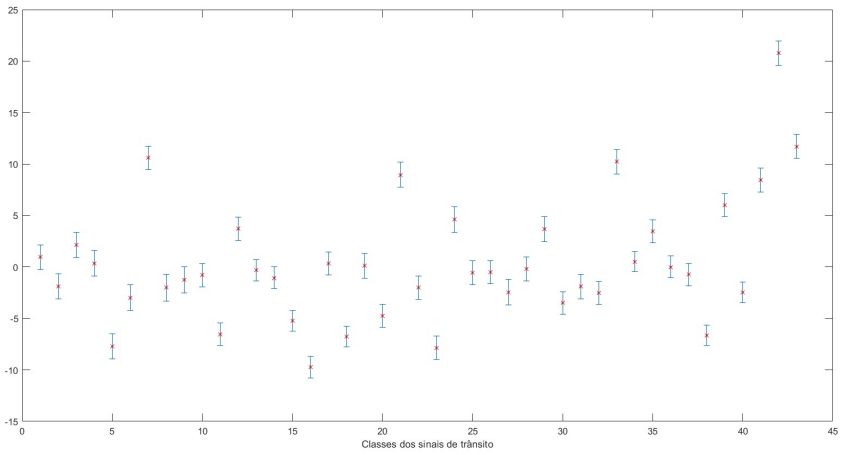

Figura 5.65: Gráfico de comparação dos conjuntos alcançáveis para cada classe a partir da imagem de entrada de classe 41 com um $\epsilon = 0.5$. O eixo das abcissas representa a classe do sinal de trânsito e o eixo das ordenadas representa os "logits" de alcançabilidade na saída da camada totalmente conectada que precede a camada de ativação Softmax.

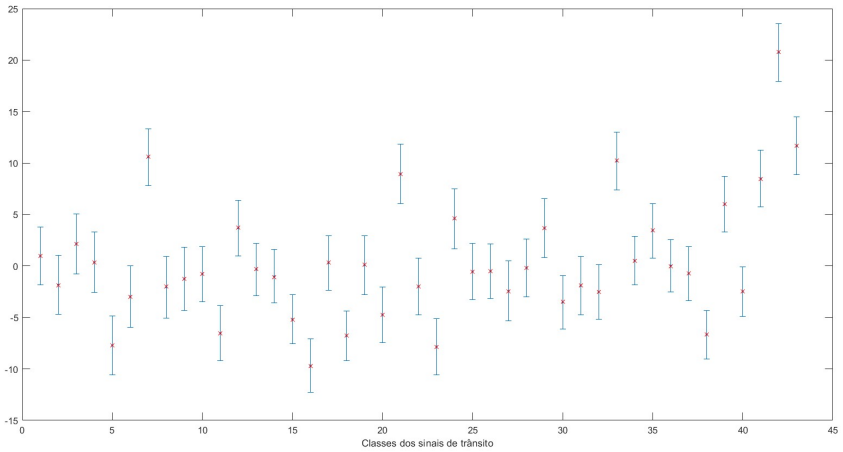

Figura 5.66: Gráfico de comparação dos conjuntos alcançáveis para cada classe a partir da imagem de entrada de classe 41 com um $\epsilon = 1$. O eixo das abcissas representa a classe do sinal de trânsito e o eixo das ordenadas representa os "logits" de alcançabilidade na saída da camada totalmente conectada que precede a camada de ativação Softmax.

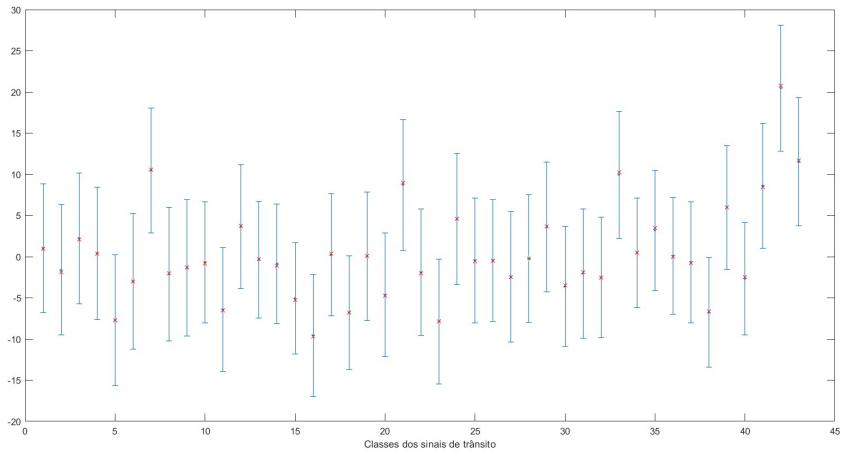

Figura 5.67: Gráfico de comparação dos conjuntos alcançáveis para cada classe a partir da imagem de entrada de classe 41 com um $\epsilon = 2$. O eixo das abcissas representa a classe do sinal de trânsito e o eixo das ordenadas representa os "logits" de alcançabilidade na saída da camada totalmente conectada que precede a camada de ativação Softmax.

5.3.2 *Discussão*

Na análise da Tabela 5.4, tal como o esperado a partir da aplicação de ataques adversariais (*i.e.* os resultados a partir do capítulo 5.2), é possível concluir que os únicos ensaios de diferentes propriedades de robustez onde o modelo CNN 1 é integralmente robusto é na imagem de teste do sinal de obrigação de sentido sempre em frente (classe 35). Relativamente às imagens de teste do sinal de proibição de exceder a velocidade máxima de 50 km/h (classe 2), sinal de perigo com curva à esquerda (classe 19) e sinal de fim de proibição de ultrapassar (classe 41), pode-se constatar para as propriedades de robustez ϕ_4 e ϕ_5, isto é, para o conjunto de perturbações limite de $\epsilon = \{2{,}3\}$, que o modelo não é robusto, apesar de o resultado gerado ser "resultado desconhecido" devido à abordagem de alcançabilidade utilizada ser a sobre-aproximada. No que se refere ao tempo de execução do processo de verificação, assim como esperado, quanto maior for o valor de perturbação limite estipulado, maior será o conjunto de valores de entrada, e por isso, maior será o tempo de cálculo de verificação formal para obtenção dos conjuntos de alcançabilidade de cada uma das 43 classes da base de dados GTSRB. Em relação aos resultados de verificação, também é possível denotar que para os 20 ensaios experimentais realizados, apenas 2 deles (10%), não conseguiram obter nenhum resultado relativamente à verificação de robustez do modelo tendo em consideração o intervalo de tempo máximo permitido de 21600 segundos (6 horas), tal como ocorre na VNN-COMP.

A partir das 18 figuras geradas pela ferramenta NNV relativas aos gráficos de comparação dos conjuntos alcançáveis finais obtidos em cada classe como forma

de análise da alcançabilidade do modelo, é possível visualizar que quanto maior for o valor de perturbação limite definido na propriedade de robustez, maior será o intervalo do conjunto de valores de entrada e por consequência, maior será o intervalo do conjunto de alcançabilidade final. Também é demonstrado que, quando algum intervalo do conjunto de alcançabilidade final de duas classes diferentes se intersetam, tal como acontece nas Figuras 5.53, 5.54, 5.58 e 5.67, então o NNV gera logo a mensagem a indicar que o resultado é desconhecido.

Uma das razões para a grande disparidade dos resultados em diferentes imagens de teste para as mesmas propriedades de robustez, pode-se dever ao facto de na base de dados GTSRB, existir um desequilíbrio na proporção do número de imagens que existe em cada classe, (*vide* capítulo 4.2), o que leva a um grupo de imagens ser robusto até um determinado valor limite de perturbação, e outro grupo de imagens ser robusto até outro valor limite de perturbação diferente.

Esta página foi propositadamente deixada em branco.

6

CONCLUSÕES E TRABALHO FUTURO

O trabalho apresentado tem como objetivo investigar e desenvolver técnicas de verificação formal de redes neuronais profundas no contexto da condução autónoma, com ênfase na propriedade de robustez frente a ataques adversariais. Este trabalho permitiu explorar e aplicar conceitos inovadores e tecnologias fundamentais, especialmente as redes neuronais convolucionais, para a classificação de sinais de trânsito, visando garantir uma maior segurança e confiabilidade nos sistemas de condução autónoma.

Os resultados experimentais demonstraram que a aplicação da verificação formal pode aumentar significativamente a robustez das redes neuronais profundas contra perturbações adversariais, o que comprova a pertinência da sua aplicação no domínio do setor automóvel. As 3 redes neuronais desenvolvidas foram treinadas e testadas com a base de dados GTSRB, sendo capazes de classificar com elevada precisão mais de 40 tipos de sinais de trânsito diferentes. Além disso, o estudo também evidenciou o impacto de diferentes ataques adversariais baseados no cálculo do gradiente (como o FGSM e o PGD), e como as técnicas de verificação formal podem auxiliar na mitigação dos seus efeitos.

No entanto, o trabalho revelou algumas limitações que merecem atenção em estudos futuros. Em primeiro lugar, a complexidade computacional envolvida na verificação formal de redes neuronais profundas é ainda um desafio significativo, essencialmente quando se considera a escalabilidade para redes de maior dimensão ou modelos mais complexos. A limitação de ferramentas disponíveis em MATLAB para verificação formal também foi um fator que restringiu algumas abordagens possíveis, sugerindo a necessidade de um desenvolvimento de bibliotecas e ferramentas mais robustas e adaptáveis neste tipo de linguagem. Da mesma forma, a natureza adversarial dos ataques analisados também revelaram a vulnerabilidade inerente que os modelos de redes neuronais atuais possuem, o que reforça a importância de metodologias preventivas e de defesas adversariais.

Em trabalhos futuros, é recomendável explorar várias vertentes que poderiam complementar e melhorar as análises realizadas. Uma linha promissora de investigação seria o estudo da verificação formal e da aplicação dos ataques adversariais testados na CNN 1 aos modelos CNN 2 e CNN 3, de modo a averiguar se são inerentemente mais resistentes às mesmas perturbações utilizadas, já que são modelos mais complexos e com um maior número de camadas do que o modelo CNN 1.

Adicionalmente, uma possível extensão deste trabalho seria a análise do desempenho das técnicas utilizadas em cenários reais, ou seja, através do uso de imagens com perturbações reais, para validar a eficácia dos modelos propostos em situações mais complexas, *e.g.* chuva, nevoeiro, e/ou neve. Outro ponto relevante seria a integração da técnica do treino adversarial aplicado ao modelo testado como defesa contra os ataques, de modo a verificar formalmente se o modelo é mais robusto ou proporciona um ambiente mais seguro.

Em suma, embora mais desafios e limitações persistam, o trabalho abre caminho para futuras investigações que possam levar a sistemas de condução autónoma cada vez mais seguros e confiáveis, promovendo uma evolução significativa na segurança e competência da mobilidade urbana.

BIBLIOGRAFIA

[1] D. Guidotti, "Verification of neural networks for safety and security-critical domains," 2022. [Online]. Available: https://ceur-ws.org/Vol-3345/paper10_RiCeRCa3.pdf

[2] S. Jana, Y. Tian, K. Pei, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," vol. 2018-May. IEEE Computer Society, 8 2018. [Online]. Available: https://arxiv.org/abs/1708.08559

[3] A. Haritonova and PixelPlex, "Machine learning in the automotive industry: Benefits, limitations, and best applications," 7 2023. [Online]. Available: https://pixelplex.io/blog/machine-learning-in-automotive/

[4] MathWorks, "Matlab," 2023. [Online]. Available: https://www.mathworks.com/products/matlab.html

[5] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The german traffic sign recognition benchmark: A multi-class classification competition," *Proceedings of International Joint Conference on Neural Networks*, 8 2011. [Online]. Available: https://ieeexplore.ieee.org/document/6033395https://sid.erda.dk/public/archives/daaeac0d7ce1152aea9b61d9f1e19370/published-archive.html

[6] F. Guarda and S. Lourenço, "Dem4ai - redes neuronais," 2023. [Online]. Available: https://github.com/ipleiria-robotics/Dem4AI/blob/main/1_Documentos/Dem4AI%20-%20Redes%20Neuronais.pdf

[7] IBM, "What are neural networks? | ibm," 2020. [Online]. Available: https://www.ibm.com/cloud/learn/neural-networks

[8] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, pp. 386–408, 1958. [Online]. Available: https://doi.org/10.1037/h0042519

[9] A. Albarghouthi, "Introduction to neural network verification," 9 2021. [Online]. Available: http://arxiv.org/abs/2109.10317

[10] Z. Brodtman, "The importance and reasoning behind activation functions: A critical component of neural networks illuminated," 11 2021. [Online]. Available: https://towardsdatascience.com/the-importance-and-reasoning-behind-activation-functions-4dc00e74db41

[11] P. Baheti, "Activation functions in neural networks [12 types & use cases]," 3 2021. [Online]. Available: https://www.v7labs.com/blog/neural-networks-activ

ation-functions

[12] DeepAI, "Feed forward neural network," 2024. [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/feed-forward-neural-network

[13] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: concepts, cnn architectures, challenges, applications, future directions," *Journal of Big Data 2021 8:1*, vol. 8, pp. 1–74, 3 2021. [Online]. Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00444-8

[14] IBM, "What are convolutional neural networks?" 2024. [Online]. Available: https://www.ibm.com/topics/convolutional-neural-networks

[15] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[16] M. L. Mastery and J. Brownlee, "A gentle introduction to batch normalization for deep neural networks," 12 2019. [Online]. Available: https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/

[17] Medium and K. Doshi, "Batch norm explained visually — how it works, and why neural networks need it," 5 2021. [Online]. Available: https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739

[18] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," 5 2015. [Online]. Available: http://arxiv.org/abs/1505.05424

[19] M. H. Meng, G. Bai, S. G. Teo, Z. Hou, Y. Xiao, Y. Lin, and J. S. Dong, "Adversarial robustness of deep neural networks: A survey from a formal verification perspective," 6 2022. [Online]. Available: http://arxiv.org/abs/2206.12227

[20] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, and X. Yi, "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability," 12 2018. [Online]. Available: http://arxiv.org/abs/1812.08342

[21] H. Wayne, "Why don't people use formal methods?" 1 2019. [Online]. Available: https://www.hillelwayne.com/post/why-dont-people-use-formal-methods/

[22] C. Baier and J.-P. Katoen, *System Verification*, 2008, pp. 1–18.

[23] E. Zürich, D. of Computer Science, S. Lab, and M. Vechev, "Reliable and interpretable artificial intelligence," 2020. [Online]. Available: https://www.sri.inf.ethz.ch/teaching/riai2020

[24] B. Meyer, "Soundness and completeness: with precision," 4 2019. [Online]. Available: https://bertrandmeyer.com/2019/04/21/soundness-completeness-precision/

[25] M. Sälzer and M. Lange, "Reachability in simple neural networks," in *Fundamenta Informaticae*, vol. 189. IOS Press BV, 10 2023, pp. 241–259. [Online]. Available: https://arxiv.org/abs/2203.07941

[26] J. Miguel and G. Leal, "Adversarial attacks to classification systems," 8 2022. [Online]. Available: https://estudogeral.sib.uc.pt/handle/10316/102193?mode=full

[27] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 12 2014. [Online]. Available: http://arxiv.org/abs/1412.6572

[28] J. Sen, A. Sen, and A. Chatterjee, "Adversarial attacks on image classification models: Analysis and defense," 12 2023. [Online]. Available: https://arxiv.org/abs/2312.16880

[29] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 7 2016. [Online]. Available: http://arxiv.org/abs/1607.02533

[30] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 6 2017. [Online]. Available: http://arxiv.org/abs/1706.06083

[31] O. Knagg and Medium, "Know your enemy - how you can create and defend against adversarial attacks," 1 2019. [Online]. Available: https://towardsdatascience.com/know-your-enemy-7f7c5038bdf3

[32] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," 10 2019. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8601309

[33] K. Ren, T. Zheng, Z. Qin, and X. Liu, "Adversarial attacks and defenses in deep learning," *Engineering*, vol. 6, pp. 346–360, 3 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S209580991930503X

[34] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," 10 2019. [Online]. Available: http://arxiv.org/abs/1910.07738http://dx.doi.org/10.1002/rob.21918

[35] S. International, "Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems," 4 2021. [Online]. Available: https://www.sae.org/standards/content/j3016_202104/

[36] H. Zhu, K. V. Yuen, L. Mihaylova, and H. Leung, "Overview of environment perception for intelligent vehicles," pp. 2584–2601, 10 2017. [Online]. Available:

https://ieeexplore.ieee.org/document/7857073

[37] Y. Wang, N. Dahnoun, and A. Achim, "A novel system for robust lane detection and tracking," *Signal Processing*, vol. 92, pp. 319–334, 2 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0165168411002532#f0030

[38] L. Chen, S. Lin, X. Lu, D. Cao, H. Wu, C. Guo, C. Liu, and F. Y. Wang, "Deep neural network based vehicle and pedestrian detection for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, pp. 3234–3246, 6 2021. [Online]. Available: https://www.researchgate.net/publication/351856075_Deep_Neural_Network_Based_Vehicle_and_Pedestrian_Detection_for_Autonomous_Driving_A_Survey

[39] S. Demarchi, D. Guidotti, A. Pitto, and A. Tacchella, "Formal verification of neural networks: a case study about adaptive cruise control," 05 2022.

[40] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[41] S. Pavlitska, N. Lambing, and J. M. Zöllner, "Adversarial attacks on traffic sign recognition: A survey," 7 2023. [Online]. Available: http://arxiv.org/abs/2307.08278

[42] R. Timofte, K. Zimmermann, and L. van Gool, "Multi-view traffic sign detection, recognition, and 3d localisation," in *Ninth IEEE Computer Society Workshop on Application of Computer Vision*, Snowbird, Utah, USA, December 2009, pp. 1–8.

[43] A. Mogelmose, M. M. Trivedi, and T. B. Moeslund, "Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1484–1497, 2012.

[44] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, "Traffic-sign detection and classification in the wild," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[45] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," 4 2018. [Online]. Available: https://arxiv.org/abs/1707.08945

[46] X. Yang, W. Liu, S. Zhang, W. Liu, and D. Tao, "Targeted physical-world attention attack on deep learning models in road sign recognition," 8 2021. [Online]. Available: https://arxiv.org/abs/2010.04331

[47] Y. Zhong, X. Liu, D. Zhai, J. Jiang, and X. Ji, "Shadows can be dangerous: Stealthy and effective physical-world adversarial attack by natural phenomenon," 3 2022. [Online]. Available: https://arxiv.org/abs/2203.03818

[48] J. Liu, B. Lu, M. Xiong, T. Zhang, and H. Xiong, "Adversarial attack with raindrops," 7 2023. [Online]. Available: https://arxiv.org/abs/2302.14267

[49] D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber, "Multi-column deep neural network for traffic sign classification," *Neural Networks*, vol. 32, pp. 333–338, 8 2012. [Online]. Available: https://ieeexplore.ieee.org/document/6248110

[50] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, vol. 32, pp. 323–332, 8 2012. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0893608012000457

[51] A. Kherraki, M. Maqbool, and R. E. Ouazzani, "Robust traffic signs classification using deep convolutional neural network," in *2022 International Conference on Intelligent Systems and Computer Vision, ISCV 2022*. Institute of Electrical and Electronics Engineers Inc., 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9806122

[52] R. Kumar and S. Agarwal, "Convolutional neural network for traffic sign classification," *International Journal of Information Technology Project Management*, 12 2022. [Online]. Available: https://www.researchgate.net/publication/366594178_Convolutional_Neural_Network_for_Traffic_Sign_Classification

[53] C. Ferencz and M. Zöldy, "Neural network-based multi-class traffic-sign classification with the german traffic sign recognition benchmark," *Acta Polytechnica Hungarica*, vol. 21, pp. 203–220, 2024. [Online]. Available: https://www.researchgate.net/publication/379072953_Neural_Network-based_Multi-Class_Traffic-Sign_Classification_with_the_German_Traffic_Sign_Recognition_Benchmark

[54] C. Berghoff, P. Bielik, M. Neu, P. Tsankov, and A. von Twickel, "Robustness testing of ai systems: A case study for traffic sign recognition," 8 2021. [Online]. Available: http://arxiv.org/abs/2108.06159http://dx.doi.org/10.1007/978-3-030-79150-6_21

[55] C. Brix, S. Bak, C. Liu, and T. T. Johnson, "The fourth international verification of neural networks competition (vnn-comp 2023): Summary and results," 12 2023. [Online]. Available: https://sites.google.com/view/vnn2023/homehttps://docs.google.com/presentation/d/1MSYvXoLHjyXW_TKeXPIEZKV4674sE2dj9h9vHzs14ak/edit?usp=sharinghttps://arxiv.org/abs/2312.16760

[56] H.-D. Tran, X. Yang, D. M. Lopez, P. Musau, L. V. Nguyen, W. Xiang, S. Bak, and T. T. Johnson, "Nnv: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems," 4 2020. [Online]. Available: https://doi.org/10.24433/CO.

[57] A. Postovan and M. Eraşcu, "Benchmarking local robustness of high-accuracy binary neural networks for enhanced traffic sign recognition," in *Electronic Proceedings in Theoretical Computer Science, EPTCS*, vol. 389. Open Publishing Association, 9 2023, pp. 120–130. [Online]. Available: https://web3.arxiv.org/abs/2310.03033

[58] MathWorks, "Deep network designer," 2024. [Online]. Available: https://www.mathworks.com/help/deeplearning/ref/deepnetworkdesigner-app.html

[59] D. H. Tran, D. M. Lopez, P. Musau, L. V. Nguyen, H.-D. Tran, D. M. Lopez, X. Yang, W. Xiang, and T. T. Johnson, "Star-based reachability analysis of deep neural networks," 2019. [Online]. Available: https://www.researchgate.net/publication/333759458

[60] G. Singh, T. Gehr, M. Püschel, and M. Vechev, "An abstract domain for certifying neural networks," *Proceedings of the ACM on Programming Languages*, vol. 3, 1 2019.

[61] H.-D. Tran, S. Bak, W. Xiang, and T. T. Johnson, "Verification of deep convolutional neural networks using imagestars," 4 2020. [Online]. Available: http://arxiv.org/abs/2004.05511

ANEXO A

A.1 FERRAMENTA DEEP NETWORK DESIGNER

A ferramenta *Deep Network Designer* [58] do MATLAB desempenha um papel fundamental nos processos precedentes à verificação formal, como a criação dos modelos de redes neuronais, o carregamento e pré-processamento das imagens da base de dados GTSRB e o treino dos modelos concebidos. Esta ferramenta simplifica todos esses processos devido à sua interface de fácil utilização, o que permite uma construção e personalização intuitivas de modelos de redes neuronais sem a necessidade de uma codificação extensiva. Na Figura A.1 está representada a CNN 1, tal como referenciada no capítulo 4.3, que foi desenhada a partir da utilização desta ferramenta.

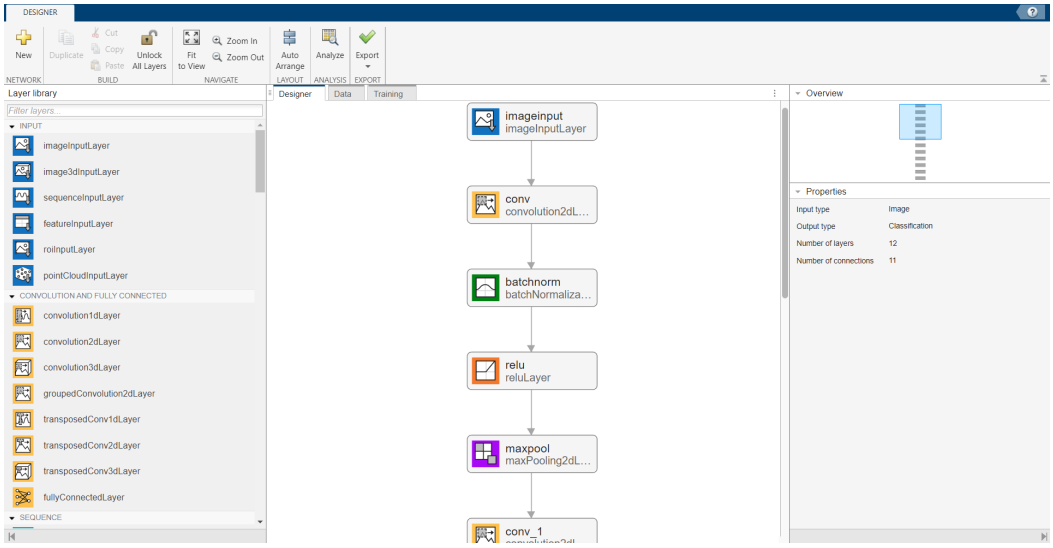

Figura A.1: Interface da ferramenta *Deep Network Designer* com uma parte do modelo de rede neuronal *CNN 1* representado.

Um dos recursos de maior destaque do *Deep Network Designer* é a simplicidade que possui de criar diferentes camadas para o modelo de rede neuronal pretendido, isto é, a sua funcionalidade de "arrastar e soltar", o que facilita a organização e a criação de arquiteturas de redes neuronais mais complexas. Um outro recurso muito importante desta ferramenta, também é a flexibilidade que os utilizadores possuem para a elaboração de diferentes modelos de redes neuronais porque pode-se experimentar diferentes tipos de camadas, funções de ativação e outros parâmetros,

sendo que isso é extremamente valioso no contexto da classificação de sinais de trânsito, em que a arquitetura da rede pode afetar significativamente a exatidão, o número de parâmetros e a alcançabilidade do modelo.

D E C L A R A Ç Ã O

Declaro, sob compromisso de honra, que o trabalho apresentado neste projeto automóvel, com o título *" VERIFICAÇÃO FORMAL DE REDES NEURONAIS PROFUNDAS EM CONTEXTO DE CONDUÇÃO AUTÓNOMA "*, é original e foi realizado por Carlos André Macedo Gonçalves (2223293) sob orientação de Professor Doutor Luís Manuel Conde Bento (luis.conde@ipleiria.pt), Professor Doutor Diogo Nascimento Baptista (diogo.baptista@ipleiria.pt) e Professora Doutora Alexandra Nascimento Baptista (alexandra.nascimento@ipleiria.pt).

Leiria, setembro 2024

Assinado por: **Carlos André Macedo Gonçalves**
Num. de Identificação: 30661853
Data: 2024.12.20 19:31:06+00'00'

Carlos André Macedo Gonçalves