TRABALHO DE CONCLUSÃO DE CURSO

CLASSIFICAÇÃO DE TIPOS DE PAVIMENTO VIA MODELOS DE APRENDIZAGEM DE MÁQUINA E SENSORIAMENTO EMBARCADO

JOÃO PEDRO COSTA DINIZ DIAS

Brasília, 18 de dezembro de 2023

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

TRABALHO DE CONCLUSÃO DE CURSO

CLASSIFICAÇÃO DE TIPOS DE PAVIMENTO VIA MODELOS DE APRENDIZAGEM DE MÁQUINA E SENSORIAMENTO EMBARCADO

JOÃO PEDRO COSTA DINIZ DIAS

Trabalho de Conclusão de Curso submetido ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Engenheiro Eletricista

Banca Examinadora

Prof. Dr. Edson Mintsu Hung, ENE/UnB _____
Orientador

Prof. Dr. Tiago Alves da Fonseca, FGA/UnB _____
Examinador interno

Eng. Dr. Renam Castro da Silva, Dell Technologies _____
Examinador externo

FICHA CATALOGRÁFICA

DIAS, JOÃO PEDRO COSTA DINIZ
CLASSIFICAÇÃO DE TIPOS DE PAVIMENTO VIA MODELOS DE APRENDIZAGEM DE MÁQUINA E SENSORIAMENTO EMBARCADO [Distrito Federal] 2023.
xvi, 79 p., 210 x 297 mm (ENE/FT/UnB, Engenheiro, Engenharia Elétrica, 2023).
Trabalho de Conclusão de Curso - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. Aprendizagem de máquina	2. Sistemas embarcados
3. Inteligência artificial	4. Processamento digital de sinais
5. Visão computacional	6. Aprendizagem profunda
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

DIAS, J.P.C.D. (2023). *CLASSIFICAÇÃO DE TIPOS DE PAVIMENTO VIA MODELOS DE APRENDIZAGEM DE MÁQUINA E SENSORIAMENTO EMBARCADO* . Trabalho de Conclusão de Curso, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 79 p.

CESSÃO DE DIREITOS

João Pedro Costa Diniz Dias
Depto. de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

Dedicatória

Dedico este trabalho a todas as pessoas que contribuíram para a minha jornada pessoal e de graduação. Agradeço a Deus por ter me concedido a serenidade para aceitar aquilo que não posso mudar, a coragem para mudar o que me foi possível e a sabedoria para saber discernir entre as duas.

À minha tia-avó Mariluce, mulher forte e batalhadora que me deu suporte e me propiciou acesso à educação de alto nível, me ensinou sobre a vida e me formou como homem; palavras não podem descrever o que você representa para mim.

À minha avó, meus pais e meu irmão, que estiveram ao meu lado sempre que houve necessidade e formam a minha base familiar.

À minha melhor amiga e namorada Louise, que me dá suporte físico, emocional e afetivo há tantos anos, não desiste de mim mesmo nos momentos mais difíceis, e foi fundamental para a minha jornada como homem e estudante, eu te amo.

Aos meus amigos Lucas, Nuno, Juliana e Julia, que me permitiram bons momentos de diversão e amizade, e aos meus amigos Bernardo, João Victor, Paulo e Vinícius, que foram, em muitos dias, o único motivo de uma gargalhada ou brincadeira, apesar do distanciamento mais recente.

Por fim, ao meu colega de curso Andrey, uma das poucas amizades que levo da graduação.

Obrigado a todos!

"O amor é o desejo de eternidade do ser amado."

- São Tomás de Aquino

JOÃO PEDRO COSTA DINIZ DIAS

Agradecimentos

Agradeço ao meu colega Andrey, por ter me indicado a oportunidade de trabalhar neste projeto.

Aos meus colegas de graduação, estágio e TCC, Giulianno e Gustavo, pelo tempo juntos desenvolvendo este projeto. Nossas discussões, reuniões e sessões de conversa foram fundamentais para o êxito deste trabalho.

Um agradecimento especial para o meu orientador, Prof. Dr. Edson Mintsu Hung, por todo o conhecimento compartilhado nas inúmeras reuniões que tivemos, pela capacidade de orientar simultaneamente um projeto profissional e acadêmico de forma leve e sem complicações, dando sugestões quando necessárias. Certamente aquilo que aprendi com o senhor levarei para o resto da minha carreira como engenheiro.

Por fim, agradeço ao pessoal da Confederação Nacional do Transporte, em especial Fábio, Márcia, Igor, Jefferson, Alisson e Tiago, por terem me recebido tão bem e propiciado um ambiente profissional e adequado para o desenvolvimento deste projeto.

JOÃO PEDRO COSTA DINIZ DIAS

RESUMO

No contexto da Pesquisa CNT de Rodovias, deseja-se automatizar alguns processos utilizados para a classificação de tipos de pavimento. Neste sentido, este trabalho lida com esta questão, por meio do desenvolvimento de modelos de aprendizagem de máquina e sistemas embarcados. Parte-se pela construção de uma solução embarcada capaz de sensoriar sinais de vibração produzidas pelo pavimento, através de um conjunto de microcontroladores e sensores. Além disso, desenvolve-se um modelo de classificação de pavimento baseado em vídeos e imagens, utilizando dados coletados na Pesquisa CNT de Rodovias 2022, separando o tipo de pavimento em duas classes, perfeito e imperfeito, atingindo-se precisão de classificação média de 91,7%. Adicionalmente, utiliza-se os dados sensoriados pelo sistema embarcado para o desenvolvimento de um segundo modelo de classificação, baseado em sinais de vibração, o que permite uma classificação complementar do tipo de pavimento, atingindo-se precisão de classificação média de 90%. Por fim, são expostas sugestões de desenvolvimentos futuros sobre o tema, ampliando o escopo da solução fornecida.

Palavras-chave: aprendizagem de máquina, sistemas embarcados, inteligência artificial, processamento digital de sinais, visão computacional, aprendizagem profunda

ABSTRACT

In the context of the Pesquisa CNT de Rodovias research, it is desired to automate some processes used to classify pavement types. In this sense, this work deals with this issue, through the development of machine learning models and embedded systems. It starts with building an embedded solution capable of sensing vibration signals produced by the pavement, through a set of microcontrollers and sensors. Furthermore, a pavement classification model based on videos and images is developed, using data collected in the Pesquisa CNT de Rodovias 2022 research, separating the type of pavement into two classes, perfect and imperfect, achieving 91,7% classification precision. Additionally, the data sensed by the embedded system is used to develop a second classification model, based on vibration signals, which allows a complementary classification of pavement type, achieving 90% classification precision. Finally, suggestions for future developments on the topic are presented, expanding the scope of the solution provided.

Keywords: machine learning, embedded systems, artificial intelligence, digital signal processing, computer vision, deep learning

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Latinos

f_s	Frequência de amostragem	[Hz]
T_s	Período de amostragem	[s, m]
η	Taxa de aprendizagem	adm.
$f_{\text{classificação}}$	Frequência de amostragem de classificação	[fps]
$t_{\text{entre-eixos}}$	Tempo de entre-eixos	[s]
$n_{\text{janelamento}}$	Número de amostras de uma janela	[amostras]

Símbolos Gregos

$\delta(k)$	Delta de Dirac
$\delta_T(k)$	Trem de impulsos

Funções, vetores e sinais

$\boldsymbol{w}(n)$	Vetor de pesos sinápticos na n-ésima iteração
$\boldsymbol{x}(n)$	Vetor de entrada da rede neural na n-ésima iteração
$x(k)$	Sinal contínuo
$\bar{x}(k)$	Sinal amostrado
$x[k]$	Sinal discreto
$h[k]$	Resposta ao impulso
$X(f)$	Espectro de Fourier
$H(f)$	Espectro de Fourier da resposta ao impulso
$y(n)$	Resposta do *perceptron* na n-ésima iteração
$d(n)$	Resposta *ground truth* na n-ésima iteração
$e_j(n)$	Sinal de erro do j-ésimo neurônio na n-ésima iteração
$v_j(n)$	Campo induzido local do j-ésimo neurônio na n-ésima iteração
$\delta_j(n)$	Gradiente local do j-ésimo neurônio na n-ésima iteração

Operadores

$(*)$	Convolução
$\mathfrak{R}\{\cdot\}$	Parte real
$(\otimes)$	Convolução bidimensional
$\varphi(\cdot)$	Função de ativação

Siglas

AM	Aprendizagem de máquina
AP	Aprendizagem profunda
CNT	Confederação Nacional do Transporte
IA	Inteligência artificial
μC	Microcontrolador
PCR	Pesquisa CNT de Rodovias
SLIT	Sistema linear invariante no tempo
SE	Sistema embarcado
UC	Unidade de coleta

ADAM	*Adaptative movement estimation*
CNN	*Convolutional neural network*
DFT	*Discrete Fourier transform*
FFT	*Fast Fourier transform*
GAN	*Generative adversarial network*
GPS	*Global positioning system*
IDE	*Integrated development environment*
I2C	*Inter-integrated circuit communication*
MDVR	*Mobile digital video recording*
MLP	*Multi-layer perceptron*
RTC	*Real-time clock*
SPI	*Serial peripheral interface*
SGD	*Stochastic gradient descent*
SSIM	*Structural similarity index measure*
UART	*Universal asynchronous receiver-transmitter*

1 INTRODUÇÃO

.

1.1 CONTEXTO

O serviço de transporte de cargas e passageiros, tipicamente, fornece um bom indicativo do progresso econômico de uma nação (Colavite e Konishi 2015). Além disso, como destaca CNT 2022, a matriz de transporte brasileira está concentrada, de forma majoritária, no modal rodoviário, correspondendo a aproximadamente $65,0\%$ do total.

Nesse sentido, medições e estudos realizados acerca do assunto são de fundamental importância, uma vez que podem balizar investimentos, tanto público como privados, que venham a melhorar a qualidade da malha rodoviária brasileira. Mais que isso, esses estudos permitem, ao constatar como está sendo empregado parte das receitas estatais, o exercício da cidadania e da fiscalização do poder público, um importante instrumento para a melhoria da gestão pública (Marques e Almeida 2004).

Nesse contexto, a Confederação Nacional do Transporte (CNT) destaca-se como um órgão que regularmente se dedica ao estudo e à medição da qualidade da malha rodoviária brasileira, em especial por meio da Pesquisa CNT de Rodovias (PCR). Realizada anualmente desde 1995, a execução se dá por meio de pesquisadores embarcados em veículos especializados, que percorrem, durante 30 dias, rotas pré-determinadas pela equipe da CNT. São avaliadas diversas métricas, como a qualidade das sinalizações verticais e horizontais, qualidade do pavimento, geometria da via, dentre outras.

1.2 MOTIVAÇÃO E JUSTIFICATIVA

Com o desenvolvimento de novas tecnologias, torna-se importante a atualização das metodologias de pesquisa, de forma que estas possam se adequar às novas demandas de fiscalização, estudo e investimento. Assim, no âmbito das pesquisas sobre a malha rodoviária brasileira, destacam-se as evoluções nas tecnologias de sensoriamento de dados, processamento digital de sinais e aprendizagem de máquina (AM) que, se combinadas, podem produzir um resultado inovador, com aplicabilidade em futuras pesquisas.

No âmbito da PCR, deseja-se evoluir a metodologia de avaliação do pavimento, atualmente realizada de maneira manual, método muito suscetível a erros subjetivos e variações entre os próprios pesquisadores. Dessarte, para além disso, o desenvolvimento de novas soluções estado-da-arte no sensoriamento de dados, processamento digital de sinais e AM são benéficas não somente para os pesquisadores, mas também para a sociedade acadêmica como um todo, que poderá

utilizar os desenvolvimento realizados para investigar novas problemáticas nessas áreas.

Assim, a partir de 2020, a CNT buscou estabelecer contato com docentes e discentes de diversas universidades brasileiras, entre elas a Universidade de Brasília e a Faculdade de Tecnologia, almejando a incorporação dos conhecimentos acadêmicos nas metodologias da pesquisa, em especial na parte de sensoriamento, aquisição e processamento de sinais. Especificamente, foi estabelecido que o foco estaria no desenvolvimento de um sistema automático de classificação de pavimentos, que utilizaria de AM para fazer inferências acerca das condições do pavimento.

Nesse contexto, surgiu a oportunidade de aplicar conhecimentos da área de processamento digital de sinais, sistemas embarcados e aprendizagem de máquina em um projeto real e benéfico para a população brasileira, bem como desenvolver um projeto relevante e atual.

1.3 OBJETIVO E METODOLOGIA

Este trabalho propõe o desenvolvimento de tecnologias capazes de realizarem a medição da qualidade de pavimentos rodoviários de forma autônoma e embarcada, no contexto da PCR, reduzindo a subjetividade e variabilidade das análises inerentes aos pesquisadores. Para isso, planejou-se o desenvolvimento de:

1. um sistema embarcado capaz de realizar o sensoriamento de vibrações mecânicas causadas pelo pavimento;
2. um modelo de aprendizagem de máquina capaz de classificar, com base em vídeos e imagens, as condições do pavimento no qual o veículo de pesquisa trafega, e
3. um modelo de aprendizagem de máquina capaz de classificar os sinais de vibração obtidos via sensoriamento com *hardware* embarcado.

Foi aplicada uma metodologia iterativa de projeto, repetindo-se o ciclo de implementação, validação e teste, para todos os componentes e tecnologias desenvolvidas, utilizando para isso dados reais coletados em saídas de campo, bem como dados oficiais da Pesquisa CNT de Rodovias 2022 e 2023.

1.4 ESTRUTURA DO TRABALHO

O trabalho foi organizado em 6 capítulos e 1 apêndice. Uma descrição mais detalhada do conteúdo de cada seção está disposto abaixo.

- **Capítulo 2:** será dada a fundamentação teórica sobre os assuntos abordados, de forma sucinta, não se comprometendo com a cobertura integral ou pormenorizada de todas as teorias

que envolvem os tópicos mencionados. Além disso, é feita uma revisão bibliográfica de alguns artigos relevantes sobre os assuntos tratados neste trabalho, e que utilizaram estratégias similares. A ideia é que, com esse capítulo, o leitor seja capaz de compreender este trabalho em sua plenitude.

- **Capítulo 3:** será descrito o processo de construção e desenvolvimento do sistema embarcado, responsável pelo sensoriamento de sinais de vibração produzidos pelo pavimento. Os sinais captados pelo sistema serão utilizados em capítulos posteriores.
- **Capítulo 4:** será desenvolvido e justificado as escolhas feitas para a criação de um modelo de classificação de imagens, utilizado para rotular imagens de pavimento obtidas na PCR em duas classes: perfeito e imperfeito.
- **Capítulo 5:** é dado continuidade ao desenvolvimento do modelo de classificação de pavimentos, mas desta vez, utilizando uma abordagem de sinais sensoriados via sistema embarcado, utilizando na natureza intrinsecamente diferente dos dois modelos para se atingir resultados mais satisfatórios.
- **Capítulo 6:** serão feitos comentários finais acerca dos resultados obtidos, bem como serão lançadas ideias para expansão do trabalho aqui executado.
- **Apêndice:** está disposto um excerto de código em Python, que permite o cálculo da velocidade de inferência do modelo de classificação de vídeos.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo dedica-se à apresentação dos conceitos teóricos principais utilizados ao longo deste trabalho, sem comprometer-se, entretanto, com a cobertura integral ou extensivamente detalhada dos assuntos discutidos. Além disso, é feita uma revisão bibliográfica de alguns trabalhos relevantes que também lidaram com os assuntos desenvolvidos neste trabalho. Assim, espera-se que, com a exposição feita e com a bibliografia citada, o leitor seja capaz de compreender todos os aspectos teóricos concernentes ao projeto desenvolvido, retornando, ao longo da leitura e sempre que necessário, ao tópico relativo deste capítulo.

2.1 PROCESSAMENTO DE SINAIS

A área de processamento de sinais é um campo interdisciplinar da engenharia e da matemática que lida com a análise, modificação e interpretação de sinais ou dados. Ela envolve a aplicação de técnicas matemáticas, estatísticas e algoritmos para extrair informações úteis e tomar decisões com base nesses sinais. A seguir será feita uma breve exposição de alguns temas da área de processamento de sinais, considerados mais pertinentes para a correta compreensão deste trabalho.

2.1.1 Noções básicas de sinais e sistemas

Segundo Lathi e Green (2005), um sinal é um conjunto de dados ou informações. Esses dados e informações estão dispostos de forma organizada, atrelando o seu valor de intensidade a alguma referência (variável independente), seja ela o tempo, a frequência, o espaço, a corrente elétrica, o potencial elétrico, ou qualquer outro elemento, a depender da natureza do dado.

Sinais podem ser classificados de diferentes formas, mas a diferenciação entre sinal contínuo ou discreto é a mais importante e utilizada ao longo deste trabalho. Define-se um *sinal contínuo* como aquele que, para *qualquer* valor da variável independente, existe um valor de amplitude associado. Exemplos de sinais que tipicamente são considerados contínuos, em suas respectivas variáveis independentes, incluem um sinal de áudio sendo produzido por um violão, ou a intensidade luminosa do Sol. Na Figura 2.1 está disposto um sinal contínuo no tempo.

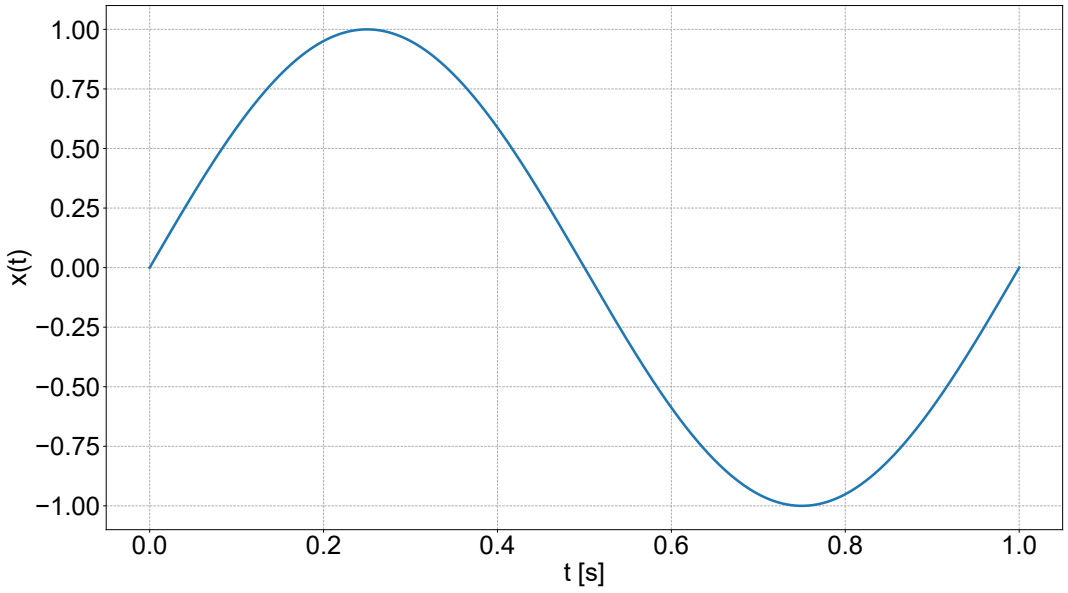


Figura 2.1: Exemplo de sinal em tempo contínuo, truncado para $0 \leq t \leq 1$

Alternativamente, define-se um *sinal discreto* como aquele que, para *determinados* valores da variável independente, existe um valor de amplitude associado, sendo que, para outros valores, a amplitude associada é indefinida. Cotações de uma ação na bolsa de valores é um exemplo de sinal em tempo discreto. Na Figura 2.2 está disposto o mesmo exemplo da Figura 2.1, mas na representação em tempo discreto.

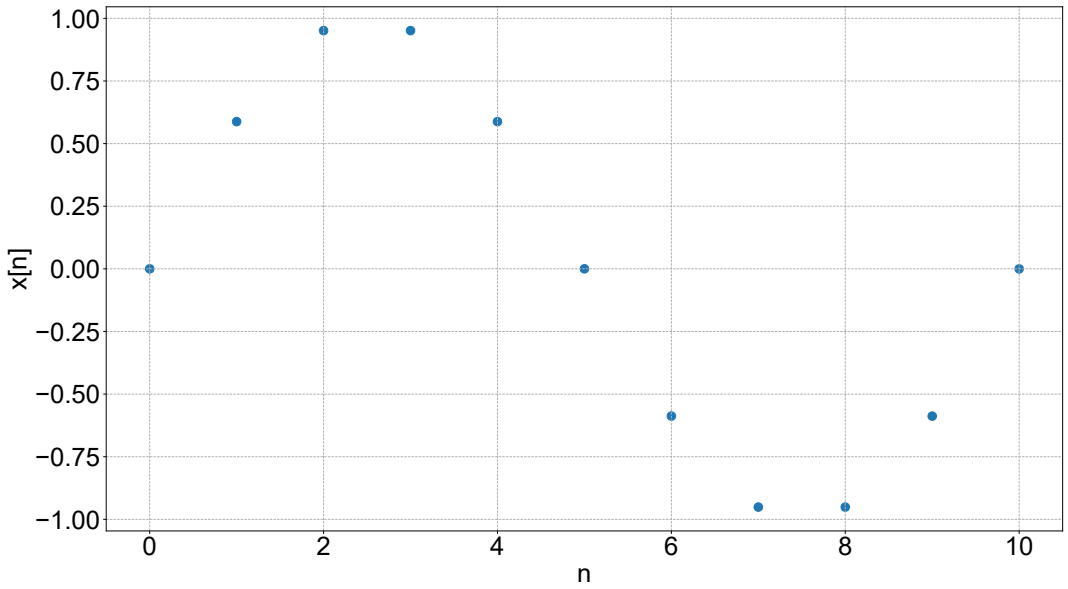


Figura 2.2: Exemplo de sinal em tempo discreto, truncado para $0 \leq n \leq 10$

Uma das grandes vantagens de sinais discretos é a capacidade de representá-los como vetores de tamanho N, onde N representa o tamanho do sinal. Assim, pode-se escrever um sinal $x[k]$ = $x[1] + x[2] + ... + x[N-1] + x[N]$ como $x[k] = (x[1], x[2], ..., x[N-1], x[N])$, uma notação muito útil quando se lida com sinais em linguagens de programação.

Ao longo deste trabalho, lidar-se-á com dados que estão atrelados às variáveis independentes *tempo*, *frequência* e *espaço*. Além disso, serão utilizados tanto sinais contínuos como discretos, para todas as variáveis independentes mencionadas.

Além do que foi discutido, encontra-se a noção de sistemas, definidos como entidades capazes de processar sinais de entrada e transformá-los em sinais de saída, com as modificações desejadas (Lathi e Green 2005). Conceitualmente, podem ser idealizados como modelos de caixa preta, onde uma entrada $x(k)$ é aplicada, sofre as modificações, e produz-se uma saída $y(k)$, como demonstra a Figura 2.3. Um exemplo típico de sistema é um equalizador musical, permitindo o balanceamento das componentes de frequência presentes em uma música, resultando em um sinal realçado ou equilibrado nas frequências desejadas.

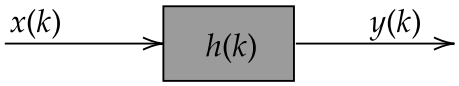


Figura 2.3: Esquemático de um sistema de caixa preta

Assim como sinais, sistemas podem ser classificados de diferentes formas. Para este trabalho, a classificação em sistema linear invariante no tempo (SLIT) é suficiente, significando que um sistema atende às propriedades de superposição e invariância temporal.

2.1.2 Resposta do sistema a uma entrada

Como mencionado anteriormente, um sistema pode ser modelado, idealmente, como um sistema de caixa preta. Tomando como referencial o esquema da Figura 2.3, a resposta de um SLIT, para uma entrada contínua, pode ser obtida fazendo

$$y(k) = x(k) * h(k)\,, \tag{2.1}$$

onde $x(k)$ representa a entrada do sistema, $h(k)$ representa a resposta ao impulso do sistema e $y(k)$ representa a saída do sistema. Além disso, a operação $x(k) * h(k)$ é a convolução entre a entrada do sistema e a resposta ao impulso, definida como

$$x(k) * h(k) \coloneqq \int_{-\infty}^{\infty} x(\tau)h(k-\tau)d\tau\,. \tag{2.2}$$

Para entradas discretas, a resposta do SLIT se torna

$$y[k] = x[k] * h[k]\,, \tag{2.3}$$

onde $x[k] * h[k]$ é a convolução discreta, definida como

$$x[k] * h[k] \coloneqq \sum_{m=-\infty}^{\infty} x[m]h[k-m]\,. \tag{2.4}$$

Uma análise intuitiva fornecida por Lathi e Green (2005) interpreta a convolução como a ponderação das componentes impulsivas do sinal original. Segundo os autores,

> "A resposta do sistema para t é, então, determinada não pela entrada $x(\tau)$, mas pelas entradas ponderadas $x(\tau)h(t-\tau)$, e o somatório de todas essas entradas ponderadas é a integral de convolução".

Essa interpretação intuitiva pode ser extendida para o caso discreto, sem perda de generalidade.

2.1.3 Transformada de Fourier

A transformada de Fourier é uma ferramenta matemática que permite obter a representação em frequência de um sinal, em termos de senoides e cossenoides complexas. Assim, pode-se avaliar qual é a composição espectral de um determinado sinal, i.e, quais frequências são mais preponderantes no sinal.

2.1.3.1 Transformada de Fourier para sinais contínuos

Para sinais contínuos, a transformada de Fourier é definida como

$$X(f) \coloneqq \int_{-\infty}^{\infty} x(k)e^{-j2\pi fk}dk\,, \tag{2.5}$$

onde k é um parâmetro que pode representar o tempo, espaço ou outras variáveis independentes.

Também é possível obter o sinal $x(k)$ a partir de $X(f)$, fazendo o cálculo da transformada inversa de Fourier, dada por

$$x(k) \coloneqq \int_{-\infty}^{\infty} X(f)e^{j2\pi fk}df\,. \tag{2.6}$$

De forma intuitiva, $X(f)$ pode ser entendido como a representação da medida de similaridade do sinal original $x(k)$ com um conjunto infinito de senoides do tipo $e^{-j2\pi fk}$.

2.1.3.2 Transformada de Fourier para sinais discretos

Para sinais discretos, a transformada de Fourier é definida como

$$X(\Omega) \coloneqq \sum_{k=-\infty}^{\infty} x[k] e^{-j\Omega k}\,, \tag{2.7}$$

onde $\Omega = \frac{2\pi}{N_0}$ representa a frequência da senóide discreta $e^{-j\Omega k}$.

Assim como o caso para sinais contínuos, é possível obter o sinal discreto original a partir da transformada de Fourier, calculando a inversa usando

$$x[k] \coloneqq \frac{1}{2\pi} \int_{2\pi} X(\Omega) e^{j\Omega k} d\Omega\,. \tag{2.8}$$

Destaca-se o fato de que, por definição, a transformada de Fourier para sinais discretos produz uma representação contínua em frequência. Além disso, pode-se provar que o espectro é 2π-periódico (Lathi e Green 2005).

2.1.3.3 Transformada discreta de Fourier (DFT)

Como o processamento de sinais discretos se dá, majoritariamente, por meio de linguagens de programação, é interessante produzir uma versão do espectro que esteja discretizada e sem informações redundantes (periódicas), facilitando o tratamento desses dados e reduzindo o custo computacional. Assim, para atender a essa demanda, surge a transformada discreta de Fourier, uma implementação do cálculo numérico da transformada de Fourier para sinais discretos.

Observando a Figura 2.4, pode-se compreender melhor a obtenção da DFT a partir do espectro de um sinal discreto. Observa-se, inicialmente, que um sinal contínuo no tempo (Fig. 2.4a) possui um espectro contínuo na frequência (Fig. 2.4b), obtido usando a transformada de Fourier para sinais contínuos, Equação (2.5). Em seguida, este sinal é discretizado (Fig. 2.4c), resultando em um espectro contínuo na frequência e periódico (Fig. 2.4d), como mencionado anteriormente, obtido pelo cálculo da transformada de Fourier para sinais discretos, Equação (2.7). Por fim, o sinal discreto é periodizado (Fig. 2.4e), resultando em um espectro discreto na frequência (Fig. 2.4f), obtido por meio do cálculo da DFT.

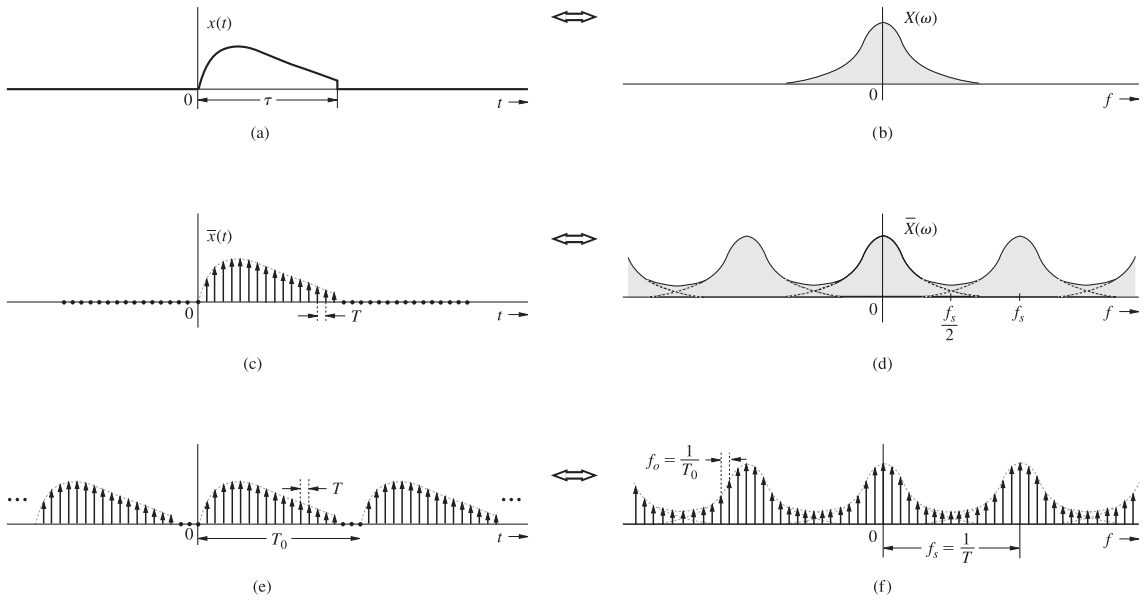


Figura 2.4: Obtenção da DFT a partir de um sinal discreto. Retirado de Lathi e Green 2005

Matematicamente, deseja-se obter o valor da r-ésima amostra do espectro discreto de Fourier do sinal, utilizando pra isso as n-ésimas amostras do sinal discreto periodizado. Assim, define-se

$$X_r \coloneqq \sum_{n=0}^{N_0-1} x_n e^{-jr\Omega_0 n}\,, \tag{2.9}$$

onde N_0 é o número de amostras em um período do sinal discreto, X_r é a r-ésima amostra da DFT e $\Omega_0 = \frac{2\pi}{N_0}$.

Adicionalmente, é possível obter a n-ésima amostra do sinal discreto original a partir da DFT, calculando a sua inversa dada por

$$x_n \coloneqq \frac{1}{N_0} \sum_{r=0}^{N_0-1} X_r e^{jr\Omega_0 n}\,. \tag{2.10}$$

Por fim, uma observação pertinente é que, por definição, $f_s = \frac{1}{T}$ e, adicionalmente, $f_0 = \frac{1}{T_0}$. Então, como $N_0 = \frac{T_0}{T}$ e $N_0' = \frac{f_s}{f_0}$, conclui-se que $N_0 = N_0'$, onde N_0' representa o número de amostras do espectro amostrado. Essa conclusão garante que o número de amostras em um período do espectro amostrado é a mesma que o número de amostras em um período do sinal original.

2.1.3.4 Transformada rápida de Fourier (FFT)

Apesar da DFT ser a solução numérica teórica para o cálculo da transformada de Fourier de sinais discretos, a sua implementação computacional mostrou-se ainda assim muito custosa. Para

solucionar esse problema, Cooley e Tukey (1965) desenvolveram o algoritmo FFT, reduzindo em muito o custo computacional do cálculo da DFT. Ao longo deste trabalho, será usada a FFT sempre que o cálculo do espectro de algum sinal discreto for realizado.

2.1.4 Filtragem de sinais

Uma das mais importantes aplicações do processamento de sinais é o processo de filtragem. Retomando o que foi descrito na Seção 2.1.2, o processo de filtragem de um sinal nada mais é do que a escolha de uma resposta ao impulso $h(k)$ adequada, produzindo um efeito desejado aos sinais de entrada.

Como descrito, ao passar um sinal de entrada pelo sistema (filtro), produz-se uma saída dada pela Equação (2.1) ou (2.3), a depender do tipo de sinal. Ademais, pode ser provado, pelo teorema da convolução, dado um sinal de entrada e a resposta ao impulso do sistema (filtro), que a amplitude do espectro da saída do sistema é dada por

$$Y(f) = X(f) \cdot H(f)\,, \tag{2.11}$$

onde $Y(f)$ é o espectro do sinal de saída, $X(f)$ o espectro do sinal de entrada e $H(f)$ o espectro da resposta ao impulso do sistema. Essa relação vale tanto para sinais de entrada contínuos como discretos (Lathi e Green 2005).

Assim, uma vez obtido o espectro da saída do sistema, é possível se determinar o sinal de saída calculando a inversa da transformada de Fourier, usando uma das várias equações dispostas anteriormente, o que elimina a necessidade do cálculo da soma ou integral de convolução entre os dois sinais.

De forma mais genérica, filtros podem ser classificados nos tipos

- Passa-baixas: permitem que frequências no intervalo $[0, f_c]$ passem, onde f_c representa uma frequência de corte definida;
- Passa-altas: permitem que frequências no intervalo $[f_c, \infty)$ passem, onde f_c representa uma frequência de corte definida;
- Passa-faixas: permitem que frequências no intervalo $[f_{c1}, f_{c2}]$ passem;
- Rejeita-faixas: rejeitam a faixa de frequência $[f_{c1}, f_{c2}]$ e permitem que o resto passe.

2.1.4.1 Componente de amplitude e fase de um filtro

Como a representação espectral de sinais nada mais é que uma combinação de senóides e cossenóides complexas, ela pode ser representada na forma polar, decomposta em uma contribuição de amplitude e uma de fase. Para perceber isso, basta relembrar que, da relação de Euler, uma

senóide do tipo $A \cdot \cos(\theta_0)$ pode ser escrita como $\Re\{Ae^{j\theta_0}\}$, donde obtêm-se a representação polar como $|A|\underline{/\theta_0}$.

Assim, ao avaliar um filtro e seus efeitos ao ser aplicado em um sinal de interesse, pode-se decompor a análise em uma componente de amplitude e outra de fase, transformando a Eq. (2.11) em

$$|Y(f)|\underline{/Y(f)} = |X(f)|\underline{/X(f)} \cdot |H(f)|\underline{/H(f)}\,. \tag{2.12}$$

Daí, obtêm-se

$$\begin{aligned} |Y(f)| &= |X(f)| \cdot |H(f)|\,, \\ \underline{/Y(f)} &= \underline{/X(f)} + \underline{/H(f)}\,. \end{aligned} \tag{2.13}$$

2.1.5 Amostragem de sinais e o teorema de Nyquist-Shannon

A maior parte dos sinais encontrados na natureza estão na forma contínua e, uma vez que atualmente o processamento de sinais se dá, majoritariamente, em computadores digitais, há a necessidade da transformação de sinais contínuos em sinais discretos. Para isso, surgiu a operação de amostragem de sinais.

O processo de amostragem de sinais nada mais é que a decomposição de um sinal em componentes impulsivas, tomadas a cada instante T_s, o período de amostragem. Para isso, matematicamente, necessita-se multiplicar o sinal original por um trem de impulsos, fazendo

$$\bar{x}(k) = x(k)\delta_T(k)\,, \tag{2.14}$$

onde $\bar{x}(k)$ representa o *sinal amostrado* e $\delta_T(k) = \sum_n \delta(k - nT)$ representa o trem de impulsos com período T. Então, a Equação (2.14) se torna

$$\bar{x}(k) = \sum_n x(nT)\delta(k - nT)\,. \tag{2.15}$$

De forma intuitiva, percebe-se que o sinal $\bar{x}(k)$ nada mais é que uma sequência discreta, composta por componentes de amplitude de $x(k)$, tomados a cada nT instantes. Efetivamente, atinge-se a discretização de um sinal originalmente contínuo.

Ademais, espera-se que o espectro de um sinal amostrado seja composto de repetições periódicas do espectro do sinal contínuo, conforme mostra a Figura 2.4. Nesse sentido, para recuperar o sinal original $x(k)$ a partir do sinal amostrado $\bar{x}(k)$, é necessário recuperar a primeira componente periódica do espectro. Assim, efetivamente, realiza-se uma filtragem passa-baixas em torno do espectro original, eliminando as componentes periódicas de frequências mais altas.

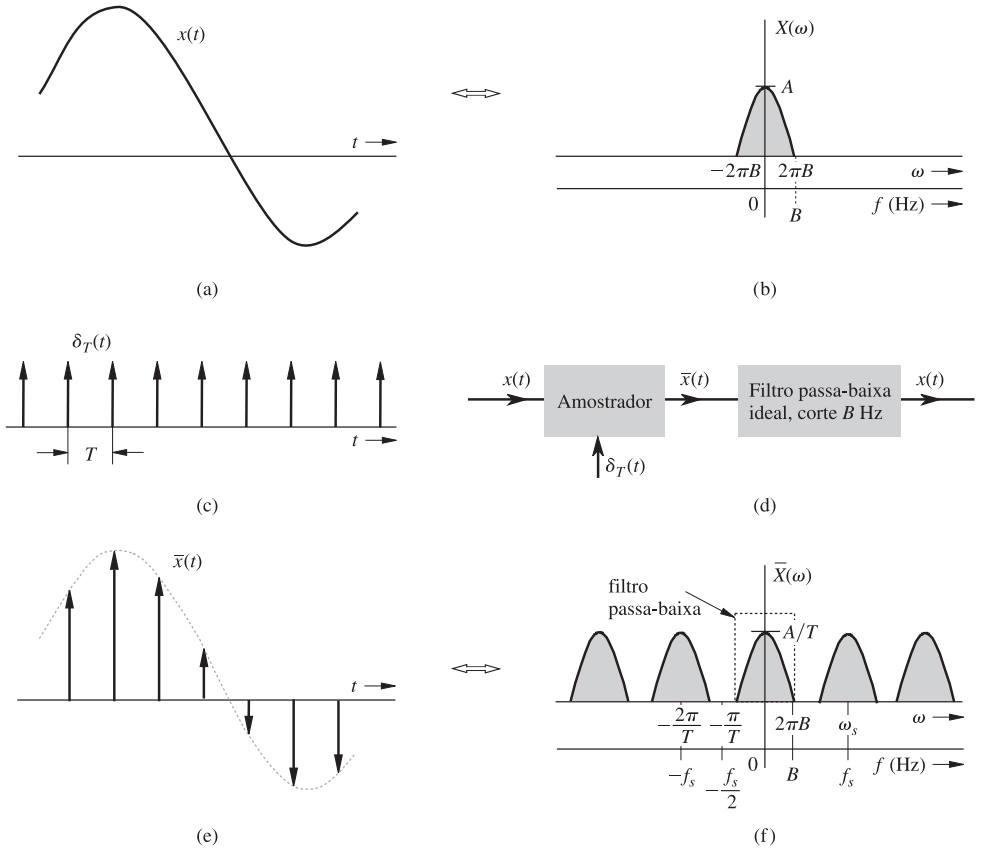


Figura 2.5: Processo de amostragem de um sinal contínuo. Retirado de Lathi e Green 2005

A Figura 2.5 mostra um processo de amostragem simplificado, onde um sinal $x(t)$ contínuo no tempo (Fig. 2.5a), de espectro $X(\omega)$ contínuo na frequência (Fig. 2.5b), é amostrado por um trem de impulsos $\delta_T(t)$ (Fig. 2.5c), como mostra o diagrama de blocos (Fig. 2.5d). Por fim, obtêm-se o sinal amostrado $\bar{x}(t)$ (Fig. 2.5e), com o respectivo espectro contínuo e periódico na frequência, filtrado para recuperar o sinal original (Fig. 2.5f).

Uma condição necessária para a correta representação de um sinal contínuo como um sinal amostrado, e para a recuperação do sinal original, é o critério de Nyquist-Shannon. Ao se observar o espectro contínuo do sinal amostrado, percebe-se que, para que não haja interferência entre os espectros espaçados periodicamente,

$$f_s > 2B\,, \tag{2.16}$$

onde B representa a largura de banda, ou a maior frequência, do sinal original (Lathi e Green 2005). Além disso, por definição,

$$T_s < \frac{1}{2B}\,. \tag{2.17}$$

Nos casos onde o critério de Nyquist-Shannon não é atendido, há a ocorrência do fenômeno de

aliasing, impossibilitando a recuperação do sinal original a partir do sinal amostrado e produzindo efeitos negativos como a distorção do sinal amostrado.

2.1.6 Imagens como sinais discretos

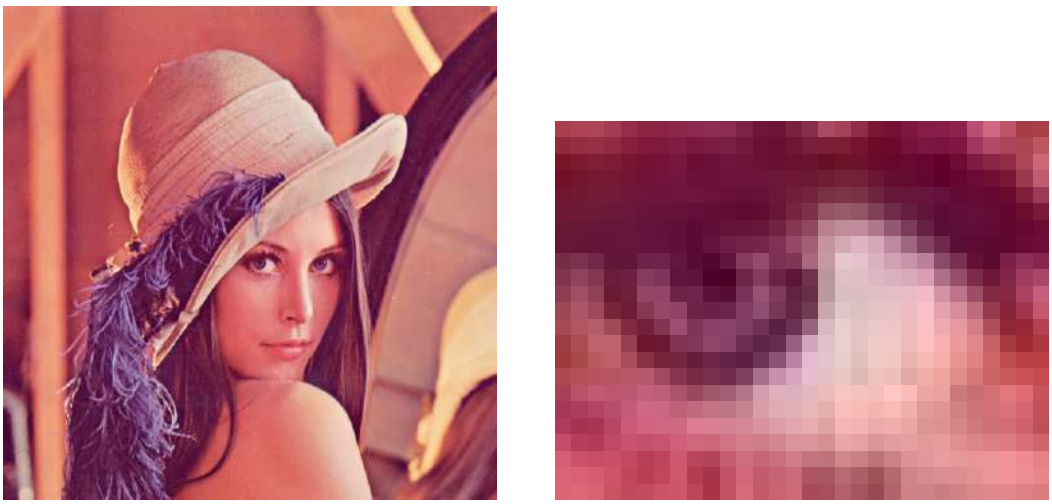
No mundo real, o olho humano percebe a realidade como sinais de intensidade luminosa contínuos no espaço, i.e., para cada coordenada no espaço (x, y) haverá uma intensidade luminosa $f(x, y)$ associada, e é isso que o olho humano percebe e o cérebro interpreta (Gonzalez 2009).

Entretanto, para processar imagens em computadores, é necessário que o conjunto infinito de coordenadas existentes em uma imagem real seja transportado para um conjunto finito de coordenadas, dentro de um limiar espacial. Assim, define-se o *pixel* como o elemento de imagem associado a uma coordenada (x, y) de uma imagem, dentro de uma possibilidade finita de coordenadas, determinada pelo tamanho da imagem, com um respectivo valor de intensidade luminosa associado.

Adicionalmente a isso, como no mundo real os seres humanos são capazes de perceber tonalidades de cores, o modelo utilizado para descrever imagens digitais deve ser capaz de refletir essa realidade. Para isso surgiram diversas soluções, sendo o modelo RGB um dos mais utilizados, que considera que toda imagem digital possui três canais, referentes às três cores fundamentais, quais sejam: vermelho (***R**ed*), verde (***G**reen*) e azul (***B**lue*). Dessa forma, imagens são compostas da concatenação desses três canais de cores fundamentais, formando a imagem colorida.

Nesse sentido, para todos os efeitos, imagens digitais podem ser entendidas como tensores $I_{c \times w \times h}$, onde cada elemento $p_{i=x,j=y,k=z}$ do tensor possui um valor de intensidade luminosa associado $f(x, y, z)$. Além disso, os valores de intensidades são limitados em magnitude, a depender da quantização realizada no momento da conversão da imagem real em imagem digital. Um intervalo de intensidade comumente utilizado é o de 8 bits, permitindo valores inteiros no intervalo $[0, 255]$, mas também são utilizadas outras quantizações, como a de 2 bits, com intervalo inteiro de $[0, 1]$ (preto e branco).

A Figura 2.6 representa um exemplo de imagem digital, composta por três canais (RGB) e quantizada com intensidade de 8 bits. Ao observar-se a imagem, não se nota nenhuma discretização nas coordenadas, de forma que ela se assemelha muito com aquilo que seria observado na realidade por meio dos olhos humanos, com uma transição suave de intensidade entre cada possível coordenada (x,y), dentro de uma possibilidade infinita de posições espaciais. Entretanto, ao aplicar-se zoom em regiões da imagem, percebe-se o efeito da discretização e da quantização da imagem, notando-se os *pixels* em cada coordenada discreta, com valores de intensidade no intervalo inteiro correspondente aos 8 bits.



(a) Imagem discretizada e digitalizada, de tamanho $3 \times 512 \times 512$ e nível de intensidade de 8 bits

(b) Zoom aplicado em torno do olho, com *pixels* perceptíveis

Figura 2.6: Exemplo de imagem digital

Nesse sentido, imagens digitais podem ser processadas como sinais discretos no espaço, seguindo as definições apresentadas anteriormente. Assim, pode-se, por exemplo, filtrar o sinal (imagem) original para que seja removido um padrão de ruído. Como descrito anteriormente, o processo de filtragem ocorre por meio de filtros selecionados, logo, no processamento de imagens digitais, a lógica é a mesma, com as adequações necessárias para a natureza bidimensional das imagens.

Dessarte, quando deseja-se filtrar espacialmente uma imagem digital, realiza-se o cálculo da convolução bidimensional entre a imagem e o filtro (também chamado de máscara), considerando apenas um canal da imagem de cada vez, ou uma imagem em tons de cinza. Matematicamente, define-se a convolução bidimensional entre uma imagem $f(x, y)$ de tamanho $w \times h$ e um filtro $w(s, t)$ de tamanho $m \times n$ como

$$(w \otimes f)(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x - s, y - t)\,, \tag{2.18}$$

onde $a = \frac{m-1}{2}$ e $b = \frac{n-1}{2}$ (Gonzalez 2009).

A Figura 2.7 fornece uma interpretação intuitiva da filtragem espacial de imagens digitais, onde a máscara $w(s, t)$ desliza sobre todos os pixels da imagem e é calculado, para cada pixel centrado na máscara, o valor da convolução bidimensional entre a máscara e a imagem.

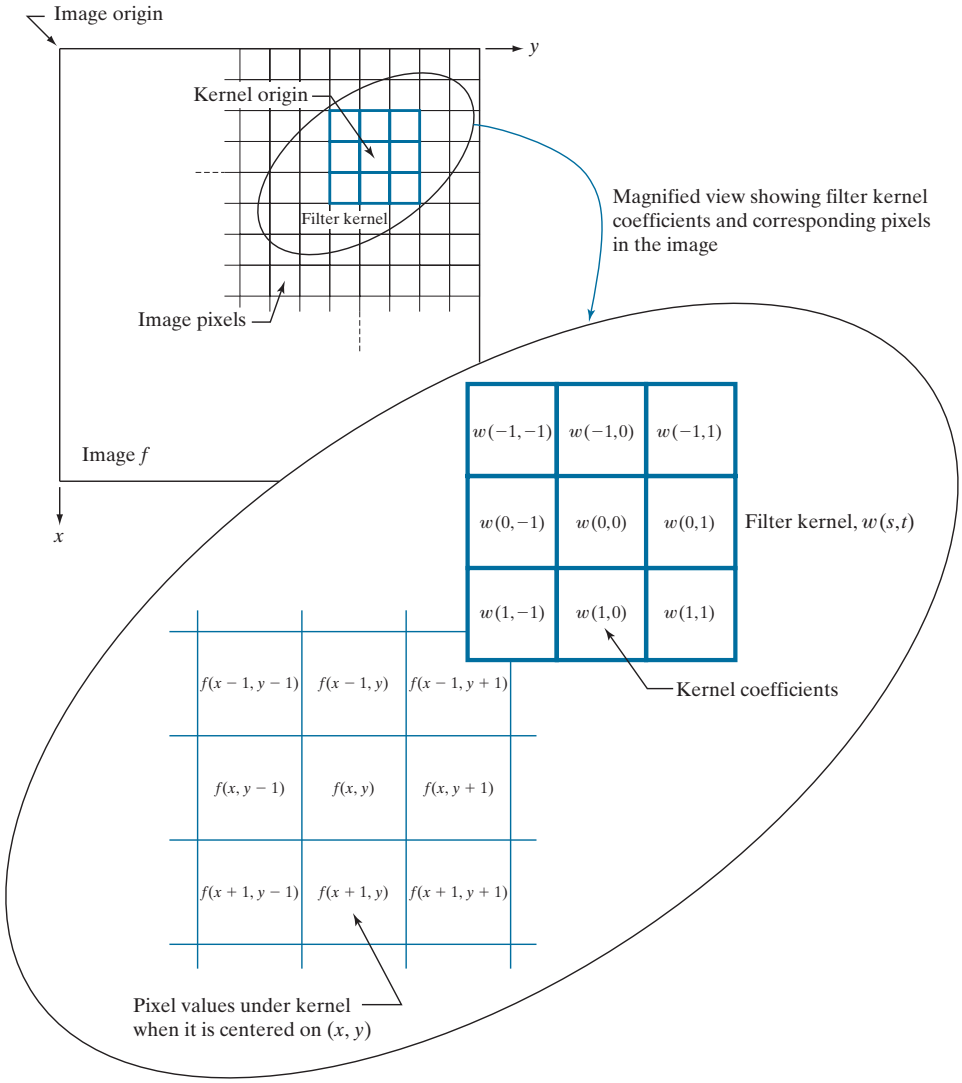


Figura 2.7: Representação da convolução bidimensional entre uma imagem e uma máscara. Retirado de Gonzalez 2009

Nesse sentido, assim como diferentes valores de respostas ao impulso produzem diferentes efeitos de filtragem em sinais unidimensionais, para imagens (sinais bidimensionais), diferentes valores de máscaras convolucionais produzem diferentes resultados. Entre os objetivos mais comuns da filtragem espacial de imagens está o de suavização, o de detecção de bordas e o de aumento de nitidez da imagem.

A título de exemplo, sejam três filtros diferentes W_1, W_2 e W_3, todos de tamanho 3×3, com valores de pesos descritos abaixo, ao realizar a convolução bidimensional entre a imagem original, disposta na Figura 2.8a, e os filtros, utilizando a Eq. (2.18), obtêm-se os resultados das Figuras 2.8b, 2.8c e 2.8d.

$$W_1 = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}, \qquad W_2 = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}, \qquad W_3 = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}.$$

(a) Imagem original em tons de cinza

(b) Suavização da imagem original usando o filtro W_1

(c) Detecção de bordas da imagem original usando o filtro W_2

(d) Aumento da nitidez da imagem original usando o filtro W_3

Figura 2.8: Imagens filtradas utilizando diferentes filtros

Alternativamente, por meio da análise de Fourier, é possível determinar o espectro da imagem e o espectro do filtro e realizar a multiplicação entre os dois espectros, realizando a filtragem desejada, conforme descrito na Seção 2.1.4.

Para além do exposto, o processamento de imagens digitais não está limitado à filtragem de

imagens, como também engloba algoritmos de segmentação, equalização de histograma, transformações espaciais, transformações morfológicas e outros.

2.2 SISTEMAS EMBARCADOS

O avanço tecnológico ao longo das últimas décadas permitiu, de forma sem precedentes, a miniaturização de eletrônicos, tornando corriqueira, por meio dos sistemas embarcados (SE), a utilização de equipamentos pessoais capazes de realizar tarefas que antes demandavam enorme capacidade computacional, ou que sequer eram controláveis de forma digital. Assim, atividades como a medição da frequência cardíaca, a transmissão e recepção de sinais de áudio sem fio, a medição e o controle da mistura ar/combustível em veículos automotivos, o controle remoto de televisões, o controle de máquinas de lavar, utilização de forno micro-ondas, e muitas outras, tornaram-se acessíveis à todas as pessoas, por um relativo baixo custo.

Nesse sentido, pode-se definir sistemas embarcados como sistemas de computação dedicados a realizar funções específicas em dispositivos ou sistemas maiores, e geralmente não tão flexíveis quanto os computadores de propósito geral, como *desktops* ou *laptops* (Heath 2002). Além disso, SE tipicamente contam com hardware e software dedicado, incorporado diretamente no dispositivo ou sistema que estão controlando, com o propósito específico definido.

A seguir serão abordados alguns tópicos importantes para o *design* de sistemas embarcados, em especial aqueles relevantes para a compreensão deste trabalho.

2.2.1 Arquitetura de sistemas embarcados

Ao longo do desenvolvimento da computação, diversas arquiteturas organizacionais de computadores surgiram, todas elas geralmente compostas dos mesmos elementos fundamentais, mas diferenciando-se na disposição desses componentes. Com o avanço da tecnologia, a arquitetura de von Neumann mostrou-se mais aplicável na maior parte das situações, tornando-se a arquitetura padrão de grande parte dos dispositivos computacionais modernos. A Figura 2.9 monstra um esquemático simplificado da arquitetura de von Neumann, destacando os componentes fundamentais, como mencionado. Não se exclui, entretanto, a aplicabilidade da arquitetura de Harvard, especificamente em campos de estudos mais modernos em sistemas embarcados.

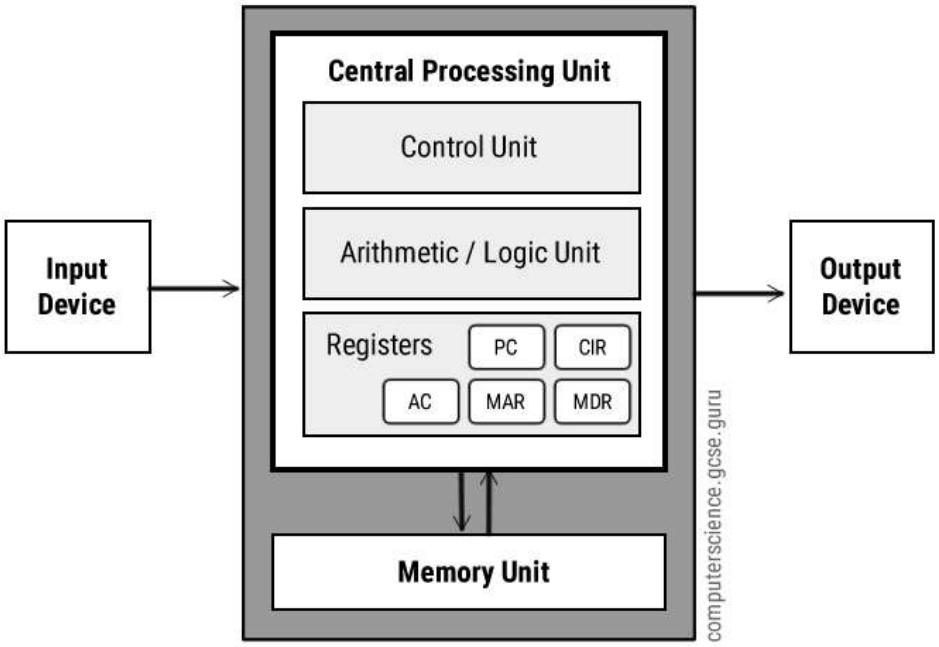


Figura 2.9: Esquemático didático da arquitetura de von Neumann. Retirado de ComputerScience.GCSE.GURU

Nesse sentido, a estrutura básica de um sistema embarcado é bastante similar a computadores de propósito geral, e consiste de um microprocessador, um sistema de memória para armazenamento de dados voláteis e não-voláteis, dispositivos de interfaceamento, periféricos e *software* necessário para executar as tarefas desejadas (Heath 2002). Dessa forma, o ponto crucial é a correta seleção dos componentes que irão compor o sistema embarcado.

Usualmente encontra-se modelos de sistemas embarcados com parte da arquitetura já definida, tipicamente o microprocessador e o sistema de memória, facilitando parte do *design* do SE, em especial para aplicações menos complexas. A esses tipos de sistemas embarcados dá-se o nome de microcontroladores, e são encontrados modelos de diversas fabricantes especializadas. Entre alguns exemplos notáveis, destacam-se os microcontroladores da família Arduino, MSP430 e ESP32.

2.2.2 Elementos sensores

Sistemas embarcados, por sua pequena dimensão e praticidade, tipicamente são empregados em aplicações que demandam o sensoriamento de dados, permitindo o processamento desses dados posteriormente ou em tempo real. Assim, para que seja possível a captação de dados do ambiente, elementos sensores são necessários.

Elementos sensores são aqueles capazes de aferir o valor de alguma variável externa e retornar o valor dessa variável para o microcontrolador, sendo alguns exemplos notáveis os sensores acelerômetro, giroscópio, termômetro, barômetro, de umidade e diversos outros. Uma vez enviado o sinal de saída do sensor, que corresponde à medição realizada, o microcontrolador realiza a inter-

pretação e o armazenamento do sinal. Como sinais encontrados no mundo real são, geralmente, contínuos, esses sensores podem optar por capturar o sinal de forma analógica e contínua e retornar esses valores para o microcontrolador por meio de uma porta de entrada analógica, ou podem já realizar a coleta de forma digital e discretizada, poupando algum trabalho do microcontrolador.

Uma característica fundamental de sensores digitais é a escolha da taxa de amostragem e da sensibilidade do sensor. O primeiro parâmetro diz respeito a frequência com a qual o sensor irá ler a variável de desejo, e a escolha do valor deve seguir os critérios estabelecidos para a correta amostragem de sinais contínuos, como explicitado na Seção 2.1.5. Já o segundo parâmetro diz respeito aos níveis de quantização do sinal, isto é, quantos bits serão necessários para representar a magnitude do sinal analógico, como mencionado na Seção 2.1.6.

2.2.3 Protocolos de comunicação serial

Para que os diferentes periféricos possam enviar e receber dados e comandos do/para o microcontrolador, é necessário que haja uma forma organizada de comunicação entre os dispositivos envolvidos, de forma que a sequência de bits (*bitstream*) seja codificada e decodificada de forma coerente por ambas as partes. Para solucionar essa questão, surgiram os denominados protocolos de comunicação serial, que estabelecem regras para o envio de dados por meio de interfaces seriais.

Para que a comunicação ocorra de forma correta, é comum definir um dispositivo como o mestre, enquanto os outros funcionam como escravos, isso é, um dispositivo é responsável por determinar quando e de qual dispositivo ele deseja receber ou enviar dados, funcionando como uma espécie de orquestrador da comunicação entre os diversos dispositivos.

2.2.3.1 *Universal Asynchronous Receiver-Transmitter* (UART)

O protocolo UART é um dos mais antigos protocolos de comunicação que ainda são utilizados. O interfaceamento se dá por meio de dois canais, denominados TX e RX, significando transmissor e receptor, respectivamente. A conexão UART só pode ser realizada entre dois dispositivos ao mesmo tempo, conectando o pino RX de um no TX do outro, e vice versa.

Por tratar-se de um protocolo assíncrono, isto é, que não possui sincronia por meio de um sinal de *clock* comum, alguns parâmetros devem ser acordados entre os dois dispositivos antes que a comunicação se inicie, como o *baud rate*, nível de tensão, bit de paridade e outros.

O pacote de dados do protocolo UART, para uma sequência de bits transmitida, segue a lógica da Tabela 2.1, onde cada coluna significa a quantidade de bits e a informação enviada nesses bits.

1	5-9	0-1	1-2
Bit de início	Bits de dados	Bit de paridade	Bits de fim

Tabela 2.1: Pacote de dados para o protocolo UART

Dessa forma, dois dispositivos configurados e conectados para comunicação UART podem utilizar dessa lógica para codificar/decodificar os dados enviados/recebidos.

2.2.3.2 *Serial Peripheral Interface* (SPI)

O protocolo SPI foi desenvolvido pela Motorola, conhecida fabricante de equipamentos eletrônicos, e trouxe como novidade a possibilidade de múltiplas conexões entre diferentes dispositivos, permitindo o controle de qual dispositivo será ativado de cada vez.

A conexão de dois dispositivos para comunicação via o protocolo SPI se dá por meio de quatro interfaces (canais), denominados

- MOSI (*Master Output, Slave Input*): saída de dados do mestre, em direção ao escravo;
- MISO (*Master Input, Slave Output*): entrada de dados do mestre, recebido do escravo;
- SCLK: sinal de *serial clock* que sincroniza os dispositivos, e
- SS/CS: sinal de *slave select* ou *chip select*, que habilita ou desabilita o dispositivo desejado.

Dessa forma, diferentemente do protocolo UART, a sinalização de início e fim de transmissão se dá por meio do sinal de CS, e os canais de dados (MOSI e MISO) são dedicados apenas para envio dos dados, permitindo o envio contínuo de bits, sem a interrupção para bits de início ou fim de sequência. Na prática, isso aumenta em muito a taxa de transmissão do protocolo SPI. Além disso, este protocolo de comunicação é do tipo síncrono, i.e., um sinal de *clock* compartilhado entre mestre e escravo garante a sincronia da comunicação, sem necessidade de sincronização prévia dos *baud rates* dos dispositivos.

Uma das maiores vantagens do protocolo SPI, em comparação com o UART, é a capacidade de múltipla conexão entre um mestre e diferentes escravos, garantida por meio dos sinais de CS, que habilita e desabilita os escravos desejados. A Figura 2.10 mostra um esquemático de configuração de comunicação com um dispositivo mestre e três escravos, mas a situação pode ser expandida para um mestre e M escravos, onde M é determinado pela capacidade máxima do canal de comunicação.

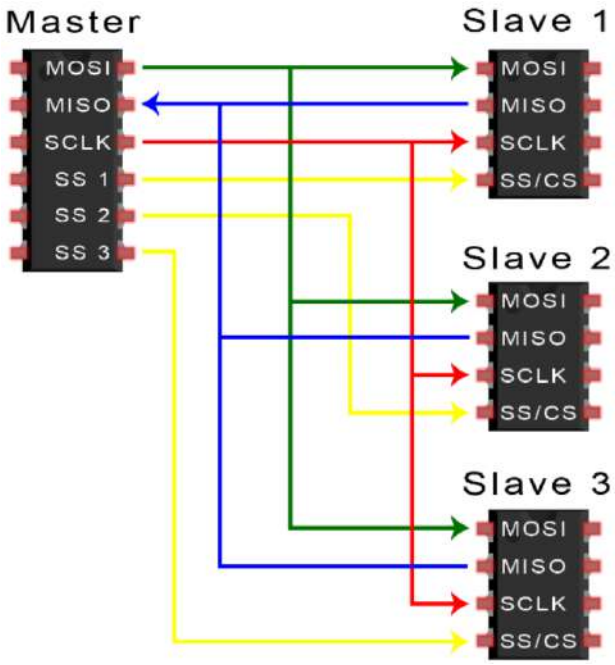


Figura 2.10: Esquemático do protocolo de comunicação serial SPI, na configuração de um mestre e três escravos

2.2.3.3 *Inter-Integrated Circuit Communication* (I2C)

O protocolo I2C é um dos mais amplamente utilizados para comunicações seriais em sistemas embarcados. Desenvolvido pela fabricante de equipamentos eletrônicos Philips em 1986, traz uma combinação das vantagens do SPI e do UART, permitindo a conexão de múltiplos dispositivos mestres com múltiplos dispositivos escravos, ao mesmo tempo que necessita apenas de dois canais para comunicação, SDA (*Serial Data Line*) e SCL (*Serial Clock Line*).

Similarmente ao protocolo UART, o protocolo I2C utiliza de pacotes de dados, seguindo a lógica disposta na Tabela 2.2, onde cada coluna significa a quantidade de bits e a informação enviada nesses bits.

	7-10	1	1	8	1		8	1	
Condição de início	Bits de endereço	Bit de leitura/ escrita	Bit de ACK/ NACK	Bits de dados	Bit de ACK/ NACK	...	Bits de dados	Bit de ACK/ NACK	Condição de fim

Tabela 2.2: Pacote de dados para o protocolo I2C

A condição de início se dá pela transição do sinal SDA de alto para baixo antes da transição do sinal SCL, e a condição de fim se dá pela transição do sinal SCL de baixo para alto antes da transição do sinal SDA. Os bits de endereço são responsáveis por determinar qual dispositivo escravo está sendo selecionado, funcionando como uma espécie de *chip select* do protocolo SPI. O bit de ACK/NACK serve como uma espécie de bit de verificação, enviado pelo receptor, do recebimento da mensagem transmitida, permitindo o re-envio de dados caso haja algum erro.

Similarmente ao UART, o protocolo I2C envia dados em pacotes, e não de forma contínua como no SPI.

2.2.4 Desenvolvimento de sistemas embarcados

A lógica de desenvolvimento de sistemas embarcados tipicamente segue alguns passos essenciais, sendo eles:

1. Delimitação do escopo de ação do SE e quais objetivos alcançar com a implementação do SE;
2. Escolha do *hardware* com capacidade computacional necessária para desempenhar as funções pretendidas do SE;
3. Escolha dos periféricos necessários para exercer as funções;
4. Desenvolvimento de *software* que implementa as funções desejadas.

Cada escolha tomada nesses passos leva em conta uma série de condições que se autoconectam, como, por exemplo, a escolha da plataforma (*hardware*) pode determinar qual linguagem de programação será usada para desenvolver o *software* do dispositivo. Nesse sentido, no processo de desenvolvimento do SE, utiliza-se da lógica disposta na Figura 2.11.

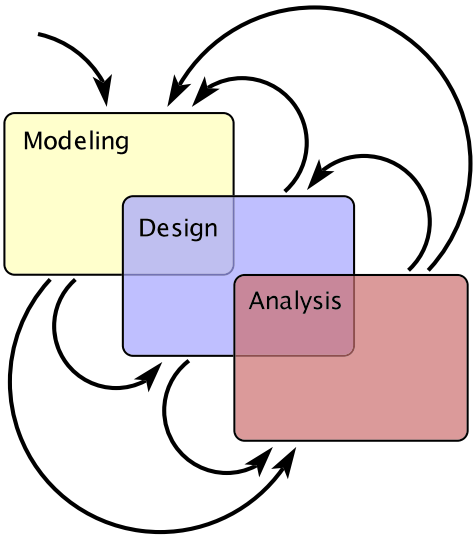


Figura 2.11: *Pipeline* de desenvolvimento de sistemas embarcados. Retirado de Lee e Seshia 2017

Assim, o desenvolvimento de sistemas embarcados é um processo iterativo de modelagem, *design* e análise (Lee e Seshia 2017). Em cada uma dessas etapas, mede-se a desempenho do sistema de acordo métricas escolhidas e, caso não esteja satisfatória para cumprir as tarefas desejadas, retorna-se à etapa anterior.

2.3 REDES NEURAIS E APRENDIZAGEM PROFUNDA

Redes neurais e aprendizagem profunda são dois pilares fundamentais no campo da inteligência artificial, que têm revolucionado o mundo atual. Essas tecnologias são responsáveis por avanços notáveis em áreas que vão desde o reconhecimento de voz e imagem até a tradução automática de idiomas, impulsionando a automação de tarefas, a personalização de serviços e a análise de grandes volumes de dados.

Neste contexto, redes neurais artificiais são modelos computacionais inspirados no funcionamento do cérebro humano, compostos por camadas de neurônios interconectados que processam informações de maneira semelhante aos circuitos neurais biológicos. A aprendizagem profunda, por sua vez, refere-se a técnicas que capacitam essas redes a aprender de forma autônoma a partir de dados brutos, em múltiplas camadas, tornando-as capazes de realizar tarefas complexas.

Nesta seção serão abordados os aspectos teóricos fundamentais dessas duas áreas, de forma que o leitor seja capaz de compreender os desenvolvimentos, que envolvam esses assuntos, realizados ao longo deste trabalho.

2.3.1 Desenvolvimentos pioneiros

Nos anos iniciais das pesquisas de redes neurais, diversos estudiosos se dedicaram ao estudo da mente humana, e como esta poderia ser esquematizada e representada em termos matemáticos. O principal foco desses esforços estava na interpretação da mente humana, em termos de subestruturas computacionais que se assemelhassem com aquilo que os pesquisadores tinham acesso. Assim, pode-se destacar algumas contribuições que impactaram significativamente os avanços nesse campo de estudo.

De acordo com Haykin (2009), entre as contribuições pioneiras, destaca-se o papel desempenhado por McCulloch e Pitts (1943), que propuseram um modelo inicial de neurônio artificial, uma das primeiras tentativas de representar o funcionamento dos neurônios biológicos em termos computacionais-matemáticos, estabelecendo as bases para a criação de redes neurais artificiais.

Além disso, destaca-se o trabalho realizado por Rosenblatt (1958), responsável pela introdução do *perceptron*, um modelo de neurônio artificial que emprega, além dos avanços previamente realizados por outros autores, o conceito de aprendizado supervisionado, permitindo que o modelo seja treinado para tomar decisões com base em dados de entrada (*inputs*) e saída conhecidos (*labels* ou *ground truth*), abrindo portas para aplicações práticas em reconhecimento de padrões e diversas outras áreas.

2.3.2 Perceptron e perceptron multicamadas

2.3.2.1 Perceptron

O modelo *perceptron* desenvolvido por Rosenblatt usa como base o modelo de neurônio de McCulloch-Pitts, sendo, como mostra a Figura 2.12, nada mais que uma combinação linear das entradas $(x_1, x_2, ..., x_m)$, ponderadas pelos pesos sinápticos $(w_1, w_2, ..., w_m)$, seguida por um limitador (função sinal).

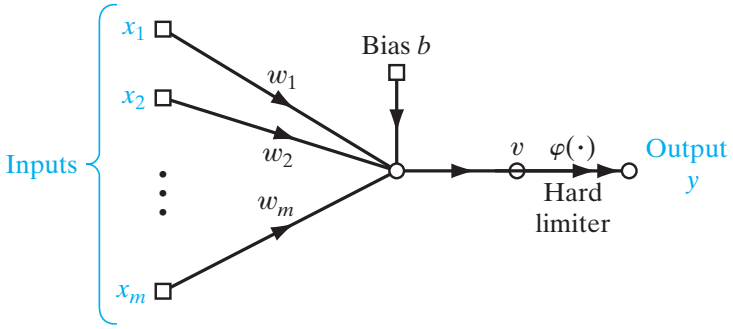


Figura 2.12: Esquemático do *perceptron*. Retirado de Haykin 2009

O objetivo do *perceptron* é a classificação do conjunto de entradas $x_1, x_2, ..., x_m$ em duas classes $\mathcal{C}_1$ e $\mathcal{C}_2$, usando para isso a saída do neurônio. Em termos matemáticos, pode-se escrever a saída y do neurônio como

$$y = \varphi(v) = \varphi\left(\sum_{i=1}^{m} w_i x_i + b\right), \tag{2.19}$$

onde v é chamado de campo induzido local e $\varphi(\cdot)$ representa uma função de ativação, neste caso a função sinal, dada por

$$\varphi(\cdot) = \text{sgn}(x) = \begin{cases} -1 & \text{se } x < 0\,, \\ 0 & \text{se } x = 0\,, \\ 1 & \text{se } x > 0\,. \end{cases} \tag{2.20}$$

Nesse sentido, se $y = +1$, o conjunto de entradas é classificado como pertencente à classe $\mathcal{C}_1$, e se $y = -1$, é classificado como pertencente à classe $\mathcal{C}_2$.

Uma das grandes contribuições trazidas pelo *perceptron* é a introdução de um algoritmo para atualização dos pesos sinápticos $(w_1, w_2, ..., w_m)$. De forma simplificada, a cada iteração do algoritmo, uma amostra é selecionada de um conjunto previamente classificado, a resposta do perceptron é calculada e a seguinte equação é utilizada para atualização dos pesos:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta[d(n) - y(n)]\mathbf{x}(n)\,, \tag{2.21}$$

onde $\mathbf{w}$ representa o vetor de pesos sinápticos, η representa o parâmetro de taxa de aprendizagem (*learning-rate*), $d(n)$ representa a resposta previamente rotulada para a amostra selecionada (*label*), $y(n)$ representa a resposta obtida do perceptron e $\mathbf{x}(n)$ representa o vetor de entrada da amostra selecionada.

Dessa forma, seja uma amostra selecionada pertencente à classe $\mathcal{C}_1$, caso a resposta $y(n)$ do perceptron, ao ser apresentada essa amostra, seja -1, ou seja, o perceptron classifique erroneamente a amostra, a Eq. (2.21) determina que o vetor de pesos sinápticos correspondente à iteração $(n+1)$ seja atualizado. Já se a resposta do perceptron for $+1$, ou seja, o perceptron classifique corretamente a amostra, a equação determina que o peso sináptico não seja atualizado. Efetivamente, este algoritmo realiza uma forma de otimização do vetor de pesos sinápticos, de forma que ele minimize o sinal de erro $[d(n) - y(n)]$.

2.3.2.2 Perceptron multicamadas

Com o avanço da teoria de redes neurais, o *perceptron* multicamadas (MLP) surgiu como uma versão mais sofisticada que, como o nome sugere, consiste da concatenação de diversos *perceptrons*, conectados entre si, formando diversas camadas neurais, como mostra a Figura 2.13.

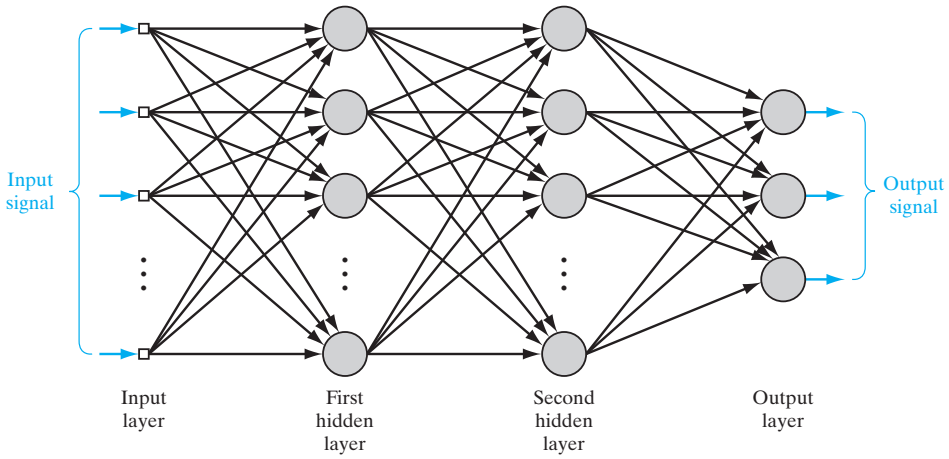


Figura 2.13: Esquemático da arquitetura de uma rede *perceptron* multicamadas. Retirado de Haykin 2009

Essa construção permite a solução de problemas de classificação mais complexos, especialmente casos de classificações não linearmente separáveis (Haykin 2009). Para a construção de um modelo multicamadas, entretanto, é necessário que algumas premissas básicas sejam atendidas, quais sejam:

- Cada neurônio da rede inclui uma função de ativação diferenciável;
- A rede inclui uma ou mais camadas que estão escondidas das entradas e das saídas;

- A rede possui um alto grau de conectividade.

Para além disso, o método de treinamento (atualização dos pesos sinápticos) ocorre em duas etapas (Silva, Spatti e Flauzino 2016):

- Fase de propagação adiante (*forward phase*): os sinais de entrada, de uma amostra do conjunto de treinamento, são propagados camada a camada até a produção das respectivas saídas.
- Fase de propagação reversa (*backward phase*): um sinal de erro é produzido ao comparar a saída da rede com uma resposta desejada. O sinal de erro resultante é propagado através da rede, novamente camada por camada, mas na direção inversa. Nesta segunda fase, ajustes sucessivos são feitos nos pesos sinápticos da rede.

O mecanismo de atualização dos pesos sinápticos dos neurônios (i.e., o mecanismo de treinamento) da rede *perceptron* multicamadas é o denominado algoritmo de retropropagação de erro (*back-propagation algorithm*), cujo funcionamento varia a depender de qual camada o neurônio a ser atualizado faz parte. Assim, Haykin (2009) subdivide o algoritmo de treinamento do perceptron em dois casos:

1. Se o neurônio j for um neurônio de saída (nó de saída) da rede, ele possui uma resposta desejável, obtida a partir da classificação prévia da amostra de treinamento fornecida à rede (dado rotulado ou *labeled data*). Dessa forma, pode-se calcular o sinal de erro $e_j(n)$, associado à essa amostra de treinamento, dado por

$$e_j(n) = d_j(n) - y_j(n)\,. \tag{2.22}$$

   Uma vez obtido o sinal de erro, calcula-se o *gradiente local* $\delta_j(n)$, usando

$$\delta_j(n) = e_j(n)\varphi'_j(v_j(n))\,, \tag{2.23}$$

   onde $\varphi'_j(v_j(n))$ representa a derivada da função de ativação do neurônio j, com respeito ao campo induzido local $v_j(n)$.

2. Se o neurônio j for um neurônio escondido (oculto), não existe um valor de resposta desejável para ele. Assim, faz-se uso da fórmula de *back-propagation* para o cálculo do gradiente local, que fornece

$$\delta_j(n) = \varphi'_j(v_j(n)) \sum_k \delta_k(n) w_{kj}(n)\,, \tag{2.24}$$

   onde $\sum\limits_k \delta_k(n) w_{kj}(n)$ representa a soma dos produtos entre o gradiente local do k-ésimo neurônio, e o peso sináptico que liga a j-ésima entrada ao k-ésimo neurônio, para os k neurônios da camada subsequente à camada j, podendo ser tanto oculta como de saída.

A interpretação intuitiva do *gradiente local* pode ser realizada imaginando uma superfície de erro tridimensional, com vales e picos locais e globais. Assim, calculando-se o vetor gradiente em um ponto qualquer desta superfície, sabe-se, da teoria de cálculo diferencial, que ele representa a direção de maior crescimento da função. Assim, na superfície de erro, como deseja-se minimizar a amplitude dessa função, toma-se a direção contrária ao vetor gradiente.

Uma vez determinados os valores de gradiente local para o neurônio j, seja ele pertencente a uma camada de saída ou a uma camada oculta, realiza-se a atualização do peso sináptico usando a Eq. (2.25).

$$\begin{pmatrix} \textit{Correção de} \\ \textit{peso sináptico} \\ \Delta w_{ji}(n) \end{pmatrix} = \begin{pmatrix} \textit{parâmetro de} \\ \textit{aprendizagem} \\ \eta \end{pmatrix} \times \begin{pmatrix} \textit{gradiente} \\ \textit{local} \\ \delta_j(n) \end{pmatrix} \times \begin{pmatrix} \textit{sinal de entrada} \\ \textit{do neurônio j} \\ y_i(n) \end{pmatrix} \tag{2.25}$$

Dessa forma, pode-se compreender o algoritmo de *back-propagation* como uma generalização do algoritmo de atualização de pesos sinápticos do *perceptron*. O algoritmo aqui apresentado foi implementado com um otimizador clássico chamado gradiente descendente, mas variações mais modernas utilizam outras opções, a citar o otimizador SGD (*stochastic gradient descent*), ADAM (*adaptative movement estimation*) e variações.

2.3.3 Redes neurais convolucionais

A arquitetura de redes neurais *perceptron* multicamadas totalmente conectadas é de fundamental importância e aplicação em diversas áreas. Entretanto, nem todo problema que envolve o uso de IA pode ser solucionado com modelos desse tipo. Especificamente no caso de imagens digitais, pela própria natureza desse tipo de sinal, surge a necessidade do desenvolvimento de arquiteturas mais sofisticadas, que levem em conta a dimensionalidade de imagens, tanto espacialmente como em termos de números de canais, como descrito na Seção 2.1.6.

Nesse sentido, procurando solucionar esta questão, LeCun et al. (1995) desenvolveu o conceito de *redes neurais convolucionais* (CNN). Essa arquitetura faz uso de conceitos do MLP, mas lança mão de três inovações, garantindo certo grau de invariância quanto a translação e distorção: campos receptivos locais, compartilhamento de pesos sinápticos e subamostragem (espacial ou temporal) (LeCun, Bengio et al. 1995).

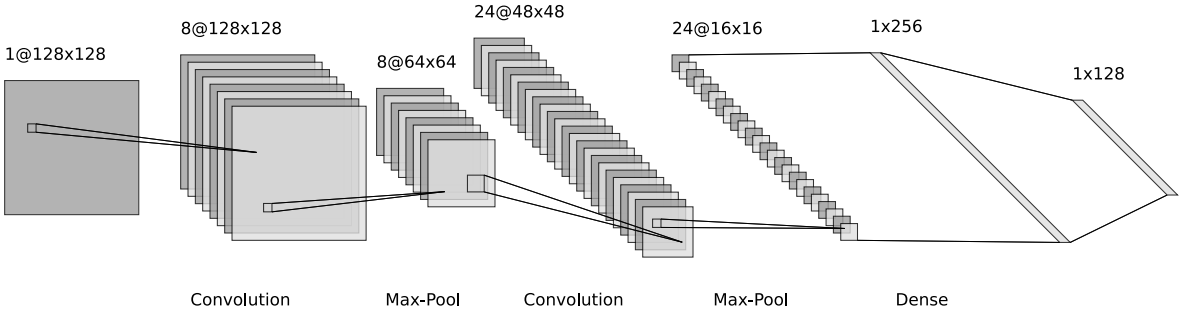


Figura 2.14: Exemplo de neural convolucional

Para ilustrar, de forma sintetizada, os conceitos das redes CNN, considera-se a Fig. 2.14. A imagem de entrada é fornecida a rede, mas, diferentemente da rede MLP – que utiliza cada pixel da imagem como entrada para um conjunto de neurônios – os sinais de entrada dos neurônios na rede CNN são compostos de pequenas regiões da imagem original, denominadas campos receptivos locais. Pode-se entender esses campos receptivos como filtros digitais de imagens, como definidos na Seção 2.1.6, onde os coeficientes do filtro correspondem aos pesos sinápticos que ligam cada entrada i a um neurônio j. Assim sendo, uma vez percorrida toda a imagem de entrada, obtém-se uma imagem de saída filtrada, denominada mapa de características (*feature map*). Este processo, na prática, permite o compartilhamento dos pesos sinápticos por todos os neurônios que compõe uma determinada camada convolucional, ou seja, que constroem um determinado mapa de características. Para cada camada convolucional, esse processo é repetido N vezes, onde N é a quantidade de mapa de características desejados, e também representa a quantidade de filtros espaciais utilizados. Como todo o processo não passa de uma filtragem espacial da imagem de entrada, os mapas de características nada mais são que características da imagem original que foram selecionadas pelo modelo, podendo corresponder a bordas, formas determinadas e outras.

Seguido à toda camada convolucional, tipicamente encontra-se uma camada de subamostragem, com o propósito de reduzir a resolução espacial da imagem, tanto para redução do custo computacional, como para aumentar a invariância espacial do modelo. Essas camadas realizam

a redução de escala da imagem de entrada, ou do mapa de características de entrada, de forma que imagens de entrada com relativa variância espacial entre si, mas pertencentes à mesma classe, consigam ser classificadas corretamente.

Ao fim das camadas convolucionais e de subamostragem, encontram-se camadas totalmente conectadas, como definidas na seção anterior. Elas realizam o papel da classificação de fato, recebendo como entrada uma versão vetorizada das características extraídas na última camada convolucional.

O processo de treinamento das redes CNN utiliza o algoritmo de *back-propagation*, definido na seção anterior. Entretanto, como os pesos sinápticos correspondem, também, a coeficientes de filtros digitais, o treinamento nada mais é que a sintetização de extratores de características (*feature extractors*), fazendo com que as características mais importantes para a classificação do conjunto de imagens sejam selecionadas. Adicionalmente, o compartilhamento de pesos sinápticos permite significativa redução do número de parâmetros treináveis, diminuindo o custo computacional (LeCun, Bengio et al. 1995).

2.3.4 Aprendizagem profunda

O conceito de aprendizagem profunda é uma extensão do conceito de redes neurais, tipicamente CNNs. Como sugere o nome, nada mais é que um modelo de ML capaz de aprender muitas características, ou seja, características profundamente intrínsecas ao problema em questão (Zhang et al. 2021). Para o caso de classificação de imagens, por exemplo, é o equivalente a dizer que um modelo de aprendizagem profunda está aprendendo a identificar aqueles aspectos mais subjetivos de cada classe, que não são facilmente perceptíveis ao olho humano.

Entre os modelos de redes neurais de aprendizagem profunda mais conhecidos, destaca-se o modelo *residual network* (*ResNet*), desenvolvido por He et al. (2016). Trata-se de um modelo de rede CNN que, ao invés de apenas aprender as características das imagens em si, aprende também o mapeamento residual das caraterísticas. A necessidade de redes residuais surge do fato de que, a partir de um certo número de camadas convolucionais, modelos CNN convencionais tendem a produzir o efeito de desaparecimento de gradiente (*fading/vanishing gradient*), bem como a aumentar enormemente o custo computacional, não se traduzindo em um aumento de precisão de classificação.

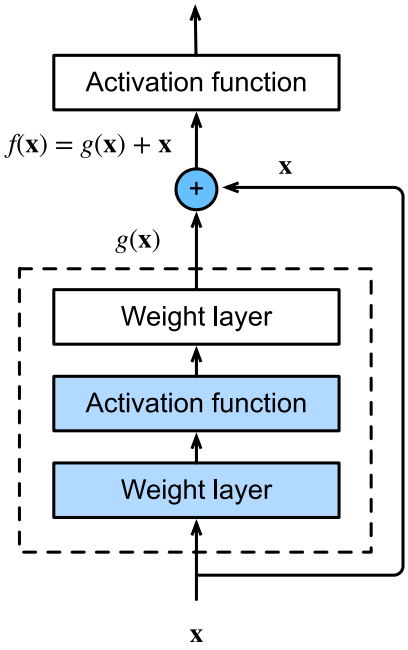


Figura 2.15: Diagrama de um bloco residual genérico. Retirado de Zhang et al. 2021

A Figura 2.15 representa um bloco residual da *ResNet*, responsável por aprender o mapeamento $g(\mathbf{x}) = f(\mathbf{x}) - \mathbf{x}$. Dessa forma, a medida que se adiciona camadas convolucionais, o modelo continua a convergir para uma solução ótima, sem o problema de desaparecimento do gradiente.

Para mais detalhes informacionais de cada camada convolucional, o tamanho dos filtros, variações da arquitetura e outros aspectos, recomenda-se a consulta do artigo original.

2.3.5 Treinamento, validação e teste de modelos de classificação

O processo de construção de um modelo de IA para classificação tipicamente passa pelas etapas de treinamento, validação e teste. Como descrito anteriormente, a natureza fundamental dos modelos de IA para classificação é a capacidade de abstrair, por meio do processo de treinamento, características que identifiquem um conjunto de entradas como pertencente a uma classe $\mathcal{C}_i$, dentre um conjunto $\mathcal{C}$ de possíveis classes. O método de treinamento de aprendizagem supervisionada utiliza um conjunto de exemplos previamente rotulados (*ground truth*), os apresenta à rede e calcula a correção dos pesos sinápticos por meio de algum algoritmo de aprendizagem, como o *back-propagation* (Haykin 2009). Outros métodos também utilizados, mas em menor frequência em modelos de classificação, é o de aprendizagem não supervisionada, onde não são apresentados exemplos rotulados manualmente ao modelo, e o de aprendizagem semi-supervisionada, onde alguns exemplos são rotulados e outros são gerados por uma rede do tipo *generative adversarial network* (GAN), por exemplo.

O processo de aprendizagem supervisionada de modelos de IA para classificação inicia-se com a construção de um *dataset*, contendo exemplos de entradas previamente rotuladas. Esse *dataset* é subdividido em três partes, a citar:

- *Dataset* de treinamento: composto por amostras rotuladas que são apresentadas ao modelo, permitindo o processo de atualização dos pesos sinápticos, para isso utilizando algum algoritmo de aprendizagem conhecido;
- *Dataset* de validação: composto por amostras rotuladas, reservadas para a obtenção de métricas de desempenho do modelo, a medida em que ele está realizando as atualizações de pesos sinápticos. O modelo não utiliza essa porção do *dataset* para propriamente atualizar os pesos, mas sim para obter métricas de desempenho do processo de treinamento do modelo;
- *Dataset* de teste: composto por amostras rotuladas, reservadas para, uma vez que o processo de treinamento esteja finalizado, obter as métricas de desempenho do modelo, verificando a capacidade de generalização do modelo. O modelo não utiliza essas imagens em nenhuma etapa de treinamento, de forma que as métricas de desempenho obtidas no teste não estejam enviesadas.

As métricas mais significativas para avaliar o desempenho de modelos de IA para classificação são a precisão de treinamento, validação e teste, e o erro de treinamento e validação. A precisão representa, para um conjunto de entrada de M exemplos classificados, quantos foram classificados corretamente pelo modelo. Já o erro é calculado a depender do critério selecionado, sendo os mais comuns o erro quadrático médio e a entropia cruzada. Tipicamente, deseja-se, com o avançar do treinamento, que a precisão aumente e o erro diminua, até atingir um platô, onde encerra-se o treinamento.

Adicionalmente, pode-se utilizar o conceito de matrizes de confusão para visualizar, de forma mais intuitiva, o comportamento do modelo para cada classe específica. Como o nome sugere, é uma matriz onde as linhas representam a classificação real de cada exemplo de teste (*label*), as colunas representam a classificação dada pelo modelo ao exemplo de teste. Os elementos da diagonal representam as classificações corretas, e qualquer elemento fora desta são classificações erradas.

2.4 REVISÃO BIBLIOGRÁFICA

A cerca do uso de modelos de aprendizagem de máquina para a classificação de defeitos do pavimento, baseado em imagens, Cao, Liu e He (2020) elaboraram uma revisão dos principais métodos estado-da-arte utilizados para tratar desse assunto. O trabalho é uma revisão dos três principais tipos de métodos usados na detecção de fissuras em pavimentos rodoviários: processamento de imagens, aprendizado de máquina e métodos baseados em imagens 3D. Os autores

discutem os diferentes algoritmos e técnicas usados em cada método, incluindo segmentação de limiar, detecção de bordas, segmentação por *region growing*, redes neurais, máquinas de vetores de suporte e aprendizagem profunda. Também são destacados os avanços na detecção de fissuras provocados por métodos de aprendizagem profunda e as pesquisas emergentes sobre detecção de fissuras usando dados 3D.

Ainda sobre esse tópico, Sholevar, Golroo e Esfahani (2022) criaram um sumário detalhado de técnicas para classificação de defeitos do pavimento, não somente utilizando vídeos e imagens, mas também dados de vibração sensoriados por acelerômetros e giroscópios. Os autores utilizaram uma abordagem individualizada para cada técnica de processamento de imagens via modelos de aprendizagem de máquina, citando referências de aplicação para casos de classificação, detecção e segmentação semântica.

Vale notar também que Chun, Yamane e Tsuzuki (2021) desenvolveram um trabalho de classificação de fissuras no pavimento, utilizando para isso uma rede do tipo ResNet50, obtendo precisão de classificação de aproximadamente 94%. Neste artigo, detalha-se, também, os limitantes da análise de defeitos do pavimento por meio de imagens de câmeras e sistemas *geographic information system* (GIS).

No tocante ao uso de dados sensoriados para classificação de defeitos do pavimento, Aboah e Adu-Gyamfi (2020) utilizaram sensores presentes em celulares *smartphones* para estimar o índice *international roughness index* (IRI). Destaca-se, também, o trabalho feito por Shtayat et al. (2023), utilizando sensor acelerômetro para a classificação de defeitos do pavimento em seis classes, obtendo precisão média de 93% ao utilizar um modelo de *random forest* e 90% utilizando um modelo do tipo *decision tree*.

Por fim, Panwar et al. (2017) utilizaram uma rede do tipo CNN para o reconhecimento de atividades humanas do dia-a-dia, como caminhar, correr, levantar, e outras. Os autores conseguiram atingir precisão de aproximadamente 99,8%, o que mostra a viabilidade da utilização de modelos de aprendizagem profunda para a classificação de dados de acelerômetro.

3 SENSORIAMENTO EMBARCADO DE VIBRAÇÕES MECÂNICAS

Este capítulo detalha as escolhas tomadas para a construção do sistema embarcado responsável pelo sensoriamento das vibrações mecânicas produzidas pelo pavimento. Objetivou-se a descrição clara e sucinta dos aspectos técnicos e práticos levados em consideração para o desenvolvimento do hardware, bem como o processo lógico de programação do software.

O primeiro objetivo delineado na introdução deste trabalho era o de construir um sistema embarcado capaz de realizar o sensoriamento de vibrações mecânicas, produzidas ao trafegar o carro da PCR nas rotas de pesquisa. Para isso, faz-se necessário um trabalho de determinação dos objetivos específicos, escolha do hardware necessário e desenvolvimento do software que irá orquestrar o sistema embarcado.

3.1 OBJETIVO ESPECÍFICO

Conforme mencionado, o objetivo primordial deste trabalho é a automatização de parte da Pesquisa CNT de Rodovias, em específico no âmbito da classificação do pavimento das rodovias avaliadas. Para compreender melhor quais são os objetivos específicos que o sistema embarcado deve atingir, deve-se explanar de forma mais clara como é classificada, atualmente, a qualidade do pavimento na PCR.

Atualmente, o pesquisador contratado pela CNT percorre, por 30 dias, diversas rotas selecionadas pela coordenação da pesquisa. Por oito horas diárias, o pesquisador, embarcado em veículos especiais, percorre as rotas e realiza anotações, a cada quilômetro, das condições predominantes do pavimento. A esses trechos de um quilômetro, dá-se o nome de unidade de coleta (UC). Seguindo os critérios da PCR, o pavimento pode ser classificado, por UC, nas seguintes categorias (CNT 2022):

1. Perfeito: neste caso, o pavimento apresenta ótima condição (sem ocorrência de defeitos) e existe perfeita regularidade na camada de revestimento. Um exemplo desse tipo de pavimento está apresentado na Figura 3.1a.
2. Desgastado: o pavimento apresenta sinais de desgaste, com efeito de desagregação progressiva do agregado da massa asfáltica e aspereza superficial no revestimento e/ou observa-se a presença de corrugação e/ou exsudação. Também pode haver isoladamente fissuras e trincas transversais ou longitudinais e remendos bem executados. Um exemplo desse tipo de pavimento está apresentado na Figura 3.1b.
3. Trincas em malha e/ou remendos: Observa-se a presença de trincas em malha e/ou remendos

mal executados. Um exemplo desse tipo de pavimento está apresentado na Figura 3.1c.

4. Afundamentos, ondulações e/ou buracos: O pavimento pode apresentar tais defeitos em conjunto ou isoladamente. Um exemplo desse tipo de pavimento está apresentado na Figura 3.1d.

5. Destruído: O pavimento apresenta elevada quantidade de buracos ou ruína total da superfície de rolamento. Neste caso, a condição da superfície do pavimento obriga os veículos a trafegarem em baixa ou baixíssima velocidade. Um exemplo desse tipo de pavimento está apresentado na Figura 3.1e.

(a) Perfeito

(b) Desgastado

(c) Trincas em malha e/ou remendos

(d) Afundamentos, ondulações e/ou buracos

(e) Destruído

Figura 3.1: Exemplos das cinco classificações utilizadas pela Pesquisa CNT de Rodovias

Dessa forma, o SE desenvolvido deve ser capaz de captar, com o maior nível possível de detalhes, as vibrações mecânicas produzidas pelo pavimento no veículo de pesquisa. Os sinais

captados pelo SE devem ser armazenados de alguma forma, permitindo que sejam encaminhados, posteriormente, para processamento via modelo de IA de classificação.

3.2 DIMENSIONAMENTO E SELEÇÃO DO HARDWARE

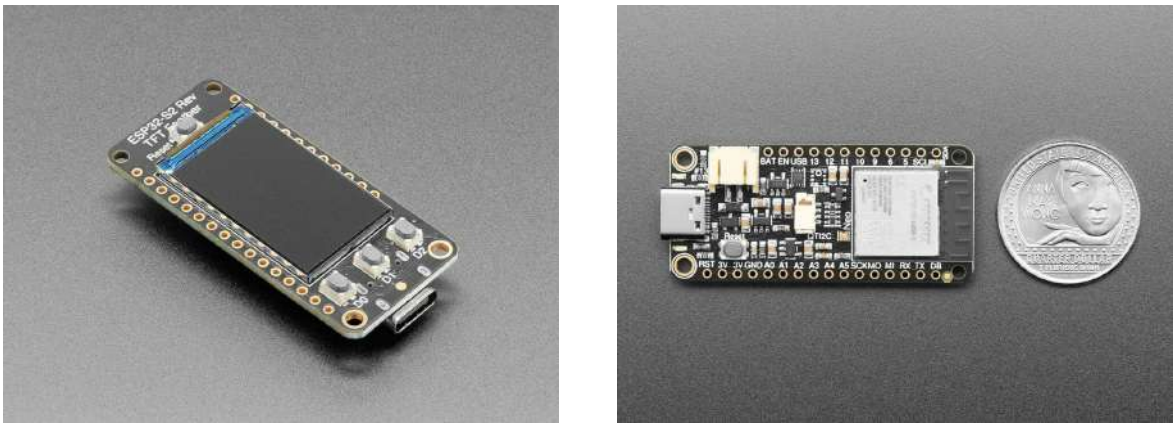
Dos objetivos desejados, concluiu-se algumas especificações necessárias para o cumprimento da missão:

1. O SE deve ser compacto e demandar pouca energia, para permitir a fácil instalação e garantir o funcionamento durante as oito horas diárias de pesquisa;
2. O SE deve contar com um elemento sensor de alta precisão, para captar os movimentos de vibração causados pelo pavimento;
3. O SE deve contar com um elemento de armazenamento, para guardar os sinais coletados ao longo da pesquisa, e permitir o uso destes sinais em modelos de classificação;
4. O SE deve possuir capacidade computacional suficiente para exercer as funções desejadas.

3.2.1 Microcontrolador

Dado essas especificações, optou-se por utilizar microcontroladores da família ESP32, da fabricante Adafruit, como orquestrador do SE. Essa escolha deveu-se, principalmente, por algumas vantagens comparativas oferecidas por esta família de μC, como a facilidade de montagem de periféricos, a disponibilidade de bibliografias, a vasta gama de opções de periféricos, o baixo custo, compactabilidade e eficiência.

Mais especificamente, selecionou-se o modelo *Adafruit ESP32-S2 Reverse TFT*, disposto na Figura 3.2. Este modelo conta com comodidades como uma tela de TFT, que facilita o processo de programação e verificação do código, bem como o conector STEMMA QT, que facilita em muito a instalação de periféricos que se comuniquem via protocolo I2C.



(a) Vista superior (b) Vista inferior

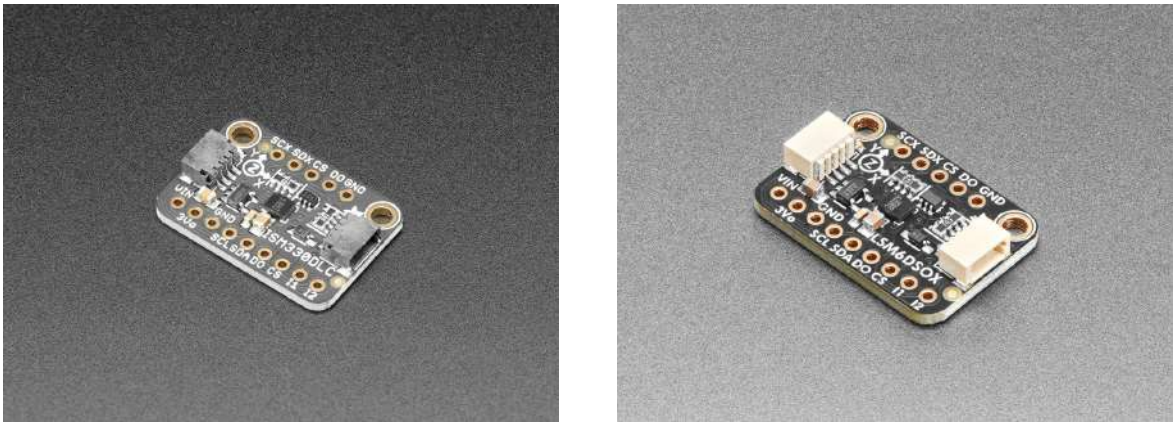
Figura 3.2: *Adafruit ESP32-S2 Revese TFT*

3.2.2 Sensor acelerômetro/giroscópio

Para o sensoriamento das vibrações, selecionou-se, inicialmente, dois modelos de acelerômetro/giroscópio: *Adafruit ISM330DHCX* e *Adafruit LSM6DSOX*. Ambos os modelos possuem conector STEMMA QT, 6 graus de liberdade e tamanho extremamente compacto. A principal diferença entre ambos os modelos é que o ISM330DHCX suporta temperaturas entre -40 e 105 graus Celsius, além de medir até ± 4000 dps no giroscópio.

Em termos de frequência de amostragem, o acelerômetro de ambos os sensores suportam a faixa de frequência discreta de $1,6$ a 6700 Hz. Já para o giroscópio, ambos os sensores suportam a faixa de frequência discreta de $12,5$ a 6700 Hz.

A escolha definitiva entre os dois modelos foi feita, posteriormente, com a análise dos sinais captados pelos dois sensores, exposto nas seções subsequentes.



(a) Adafruit ISM330DHCX (b) Adafruit LSM6DSOX

Figura 3.3: Modelos de acelerômetro/giroscópio selecionados

3.2.3 Armazenamento e comunicação

Para o armazenamento dos sinais, optou-se pelo uso de cartão SD, pela rapidez e facilidade de utilização. Para além disso, cartões SD estão disponíveis em uma infinidade de valores de capacidade, garantindo espaço suficiente para gravação de dados em um dia típico de pesquisa.

Selecionou-se o componente *Adafruit Adalogger FeatherWing*, pois apresenta, além de compatibilidade *out-of-the-box* com os outros componentes do SE, módulo de *real-time clock* (RTC) PCF8523, que permite o salvamento dos dados com o horário rotulado.

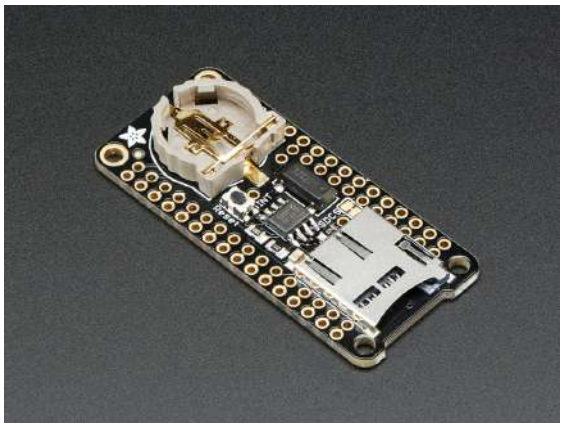


Figura 3.4: Adafruit Adalogger FeatherWing

Adicionalmente, selecionou-se um módulo Bluetooth para garantir certo grau de redundância quanto ao armazenamento de dados. Esse componente tem a função de se comunicar com o tablet de pesquisa, enviando os dados dos sensores a medida em que eles são coletados, bem como retornar o horário do tablet para o RTC, sincronizando os dados coletados pelo sistema de câmeras do veículo de pesquisa com os sinais capturados por sensores e salvos no cartão SD. Optou-se pelo modelo *JDY-31-SPP*, com capacidade Bluetooth 3.0, uma vez que o microprocessador não suporta modelos do tipo *Bluetooth low-energy*.

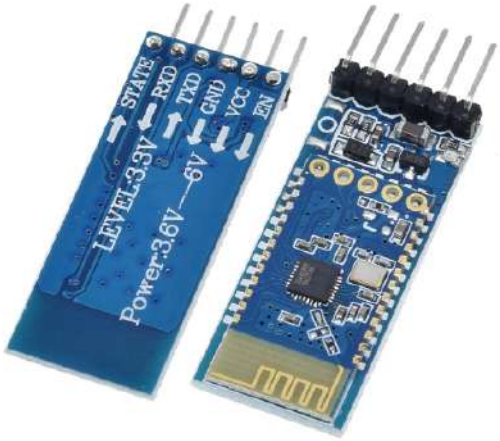


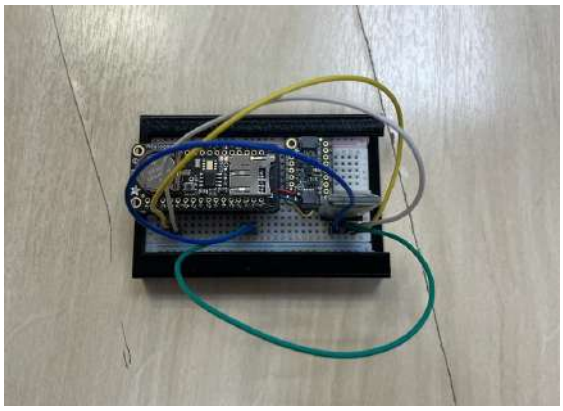
Figura 3.5: Módulo Bluetooth 3.0 JDY-31-SPP

3.2.4 Fonte de alimentação

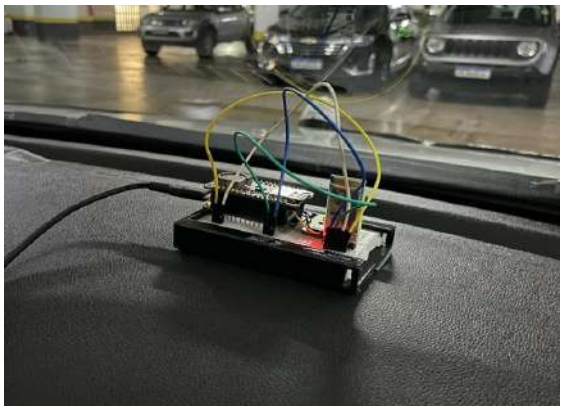
O microcontrolador conta com uma entrada do tipo USB-C, que é capaz de energizar todo o sistema e fornecer duas saída de $3,3$ V. Além disso, selecionou-se uma bateria de lítio-polímero de 500 mAh para garantir redundância ao sistema.

3.2.5 Montagem

A implementação prática do SE está disposta da Figura 3.6a, com o SE montado em uma *protoboard* com um suporte em impressão 3D, feito para fixação do SE no painel do veículo, conforme a Figura 3.6b.



(a) Sistema embarcado montado, vista superior



(b) Sistema montado e fixado no painel do veículo de pesquisa

Figura 3.6: Sistema embarcado desenvolvido

A conexão entre o microcontrolador e o armazenador de cartão SD é feito da forma "empi-

lhada", isto é, todos os pinos de ambos os componentes são conectados entre si. Já a conexão com o sensor acelerômetro/giroscópio é feita via conector STEMMA QT, uma interface I2C. Por fim, a conexão com o módulo Bluetooth é feito conectando as saída do protocolo UART do microcontrolador nos pinos TXD e RXD do módulo, além dos pinos VCC e GND.

3.3 DESENVOLVIMENTO DO SOFTWARE

Para desenvolvimento do *software* do sistema embarcado, optou-se por utilizar a linguagem de programação C++, para isso lançando mão da *integrated development environment* (IDE) do Arduino. Isso se deveu, principalmente, pela grande disponibilidade de conteúdo sobre esta linguagem e IDE, bem como bibliotecas construídas pela própria Adafruit, fabricante de maior parte dos componentes do SE, o que, além de facilitar o desenvolvimento, diminui a possibilidade de erros e garante o suporte continuado da fabricante.

Outro motivo importante para a seleção desta linguagem foi o fato dela ser do tipo compilada, isto é, o código desenvolvido é traduzido de uma só vez para linguagem de máquina e enviada ao processador, o que melhora consideravelmente o desempenho do sistema.

A lógica fundamental do algoritmo segue aquilo que foi delineado como objetivos do sistema, ou seja, ele deve: 1) aguardar uma leitura dos sensores, 2) armazenar essa leitura no cartão SD e 3) retornar ao estado de espera por uma nova leitura. A seguir está representado, na Figura 3.7, o fluxograma do algoritmo desenvolvido para captura e armazenamento dos dados sensoriados.

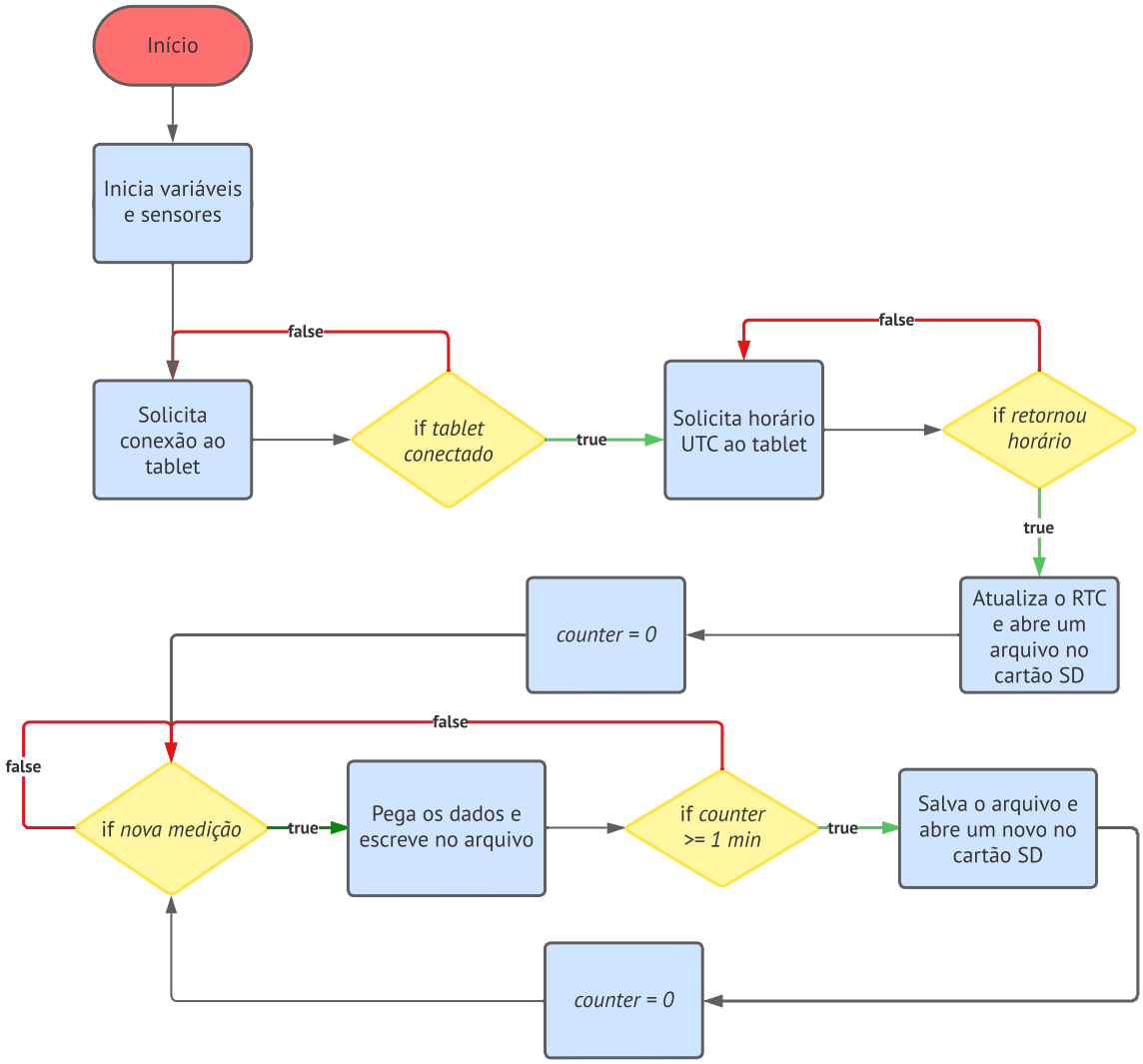


Figura 3.7: Fluxograma do algoritmo desenvolvido para captura e salvamento dos dados dos sensores acelerômetro e giroscópio

Percebe-se que foi adicionada uma condição inicial, referente ao rotulamento do horário dos dados capturados no cartão SD. Assim, o algoritmo espera, antes de tudo, que seja feita uma conexão entre o SE e o tablet de pesquisa, para, dessa forma, obter o horário UTC do tablet e sincronizá-lo no horário do RTC. Dessa forma, todas as amostras obtidas pelos sensores estarão sincronizadas com o horário do vídeo gravado, permitindo a fusão entre esses dois dados.

O *software* foi desenvolvido de forma que seja executado em tempo real, isto é, a medida em que os sensores capturam dados do ambiente, estes são salvos no cartão SD, e, a cada um minuto, o arquivo é salvo — com o horário rotulado pelo RTC — e um novo arquivo é aberto, sem perda de desempenho.

3.4 RESULTADOS OBTIDOS

Com a ideia de determinar a taxa de amostragem ideal para os sensores, realizou-se, no dia 28/07/2023, a captura de dados preliminares usando a máxima taxa de amostragem possível, de 833 Hz, limitada pela frequência de salvamento dos dados no cartão SD. O teste foi realizado com o sensor ISM330DHCX. Para captura dos dados, percorreu-se uma rota de teste de aproximadamente 20 minutos, conforme mostra a Figura 3.8.

Figura 3.8: Rota preliminar percorrida no dia 28/07/2023, para obtenção de dados de teste, marcada em amarelo

Os dados brutos dos sensores podem ser visualizados na Figura 3.9 e Figura 3.10.

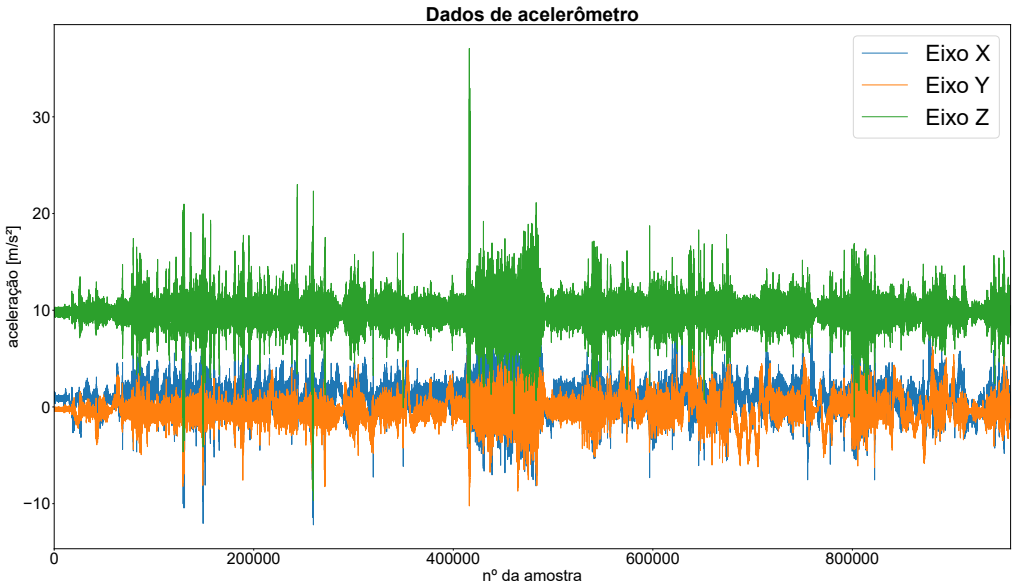


Figura 3.9: Dados brutos de acelerômetro obtidos nos testes do dia 28/07/2023

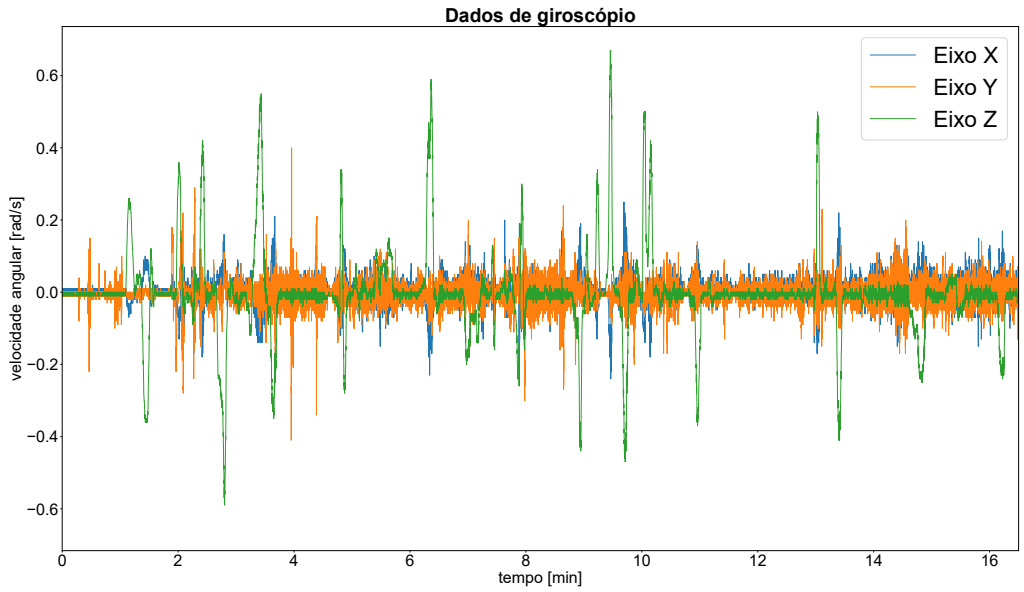


Figura 3.10: Dados brutos de giroscópio obtidos nos testes do dia 28/07/2023

Uma análise mais aprofundada das componentes de frequência do sinal mostrou que a contribuição das frequência até 30 Hz são as mais relevantes, conforme Figura 3.11. Adicionalmente, realizou-se um processo iterativo de filtragem utilizando filtro FIR passa baixas, com atenuação de 60 dB na banda de corte, iniciando com frequência de corte em torno de 90 Hz, até o valor de 30 Hz, comprovando que de fato a contribuição mais significativa do sinal está em frequências menores que 30 Hz, conforme mostra a Figura 3.12.

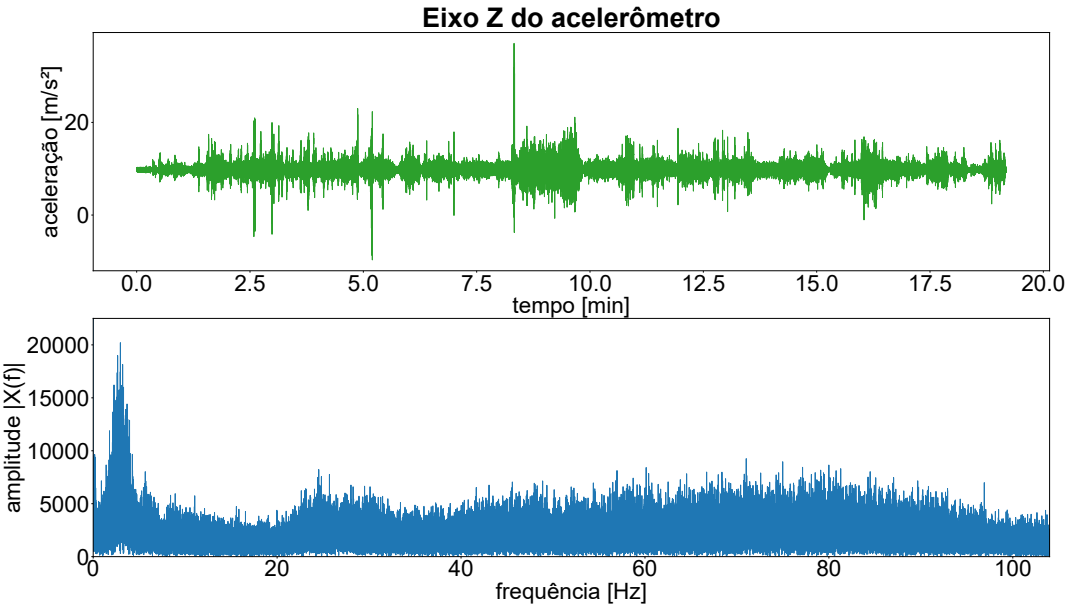


Figura 3.11: Sinal do eixo Z do acelerômetro, mostrando o espectro de Fourier do sinal

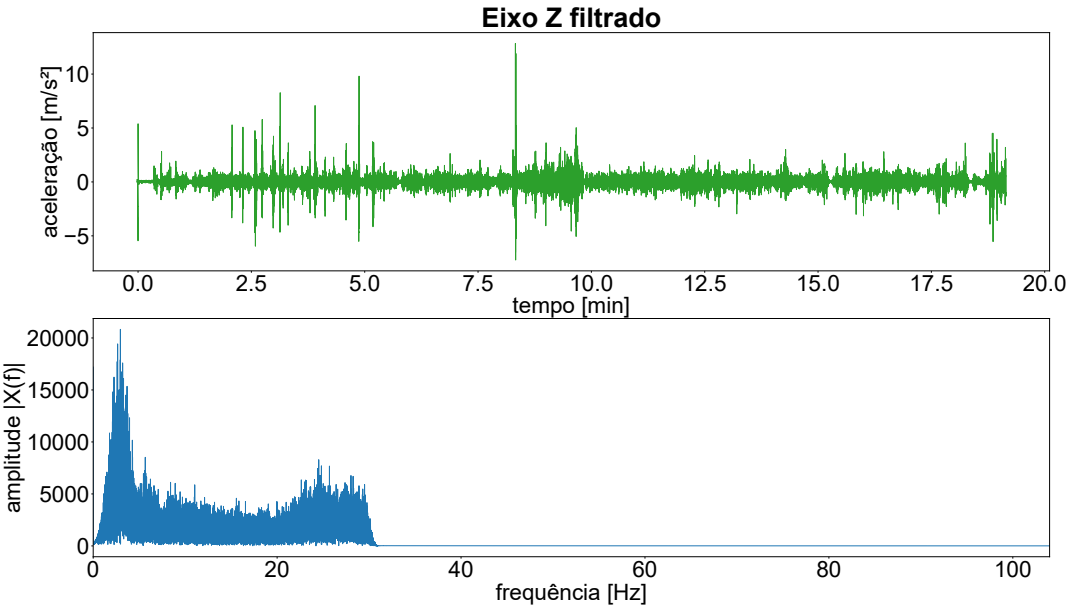


Figura 3.12: Sinal do eixo Z do acelerômetro filtrado, mostrando o espectro de Fourier do sinal filtrado

Nesse sentido, para utilização do sistema embarcado para sensoriamento das vibrações mecânicas causadas pelo pavimento, seguindo o princípio de Nyquist, uma taxa de amostragem de aproximadamente 60 Hz seria suficiente. Optou-se por utilizar a frequência de 104 Hz, que é o valor mais próximo da taxa de Nyquist que os sensores são capazes de operar.

Outro aspecto fundamental é a faixa de detecção dos sensores. Para o acelerômetro, em ambos os modelos, pode-se realizar medições de até ±16 G, isto é, ele consegue capturar valores de aceleração que sejam até 16 vezes a aceleração da gravidade. Para o giroscópio, no modelo ISM330DHCX, pode-se medir valores de velocidade angular de até 4000 dps, já para o modelo LSM6DSOX de até 2000 dps. Os testes preliminares revelaram que uma faixa de detecção de 8 G para o acelerômetro era suficiente, enquanto que para o giroscópio selecionou-se 500 dps.

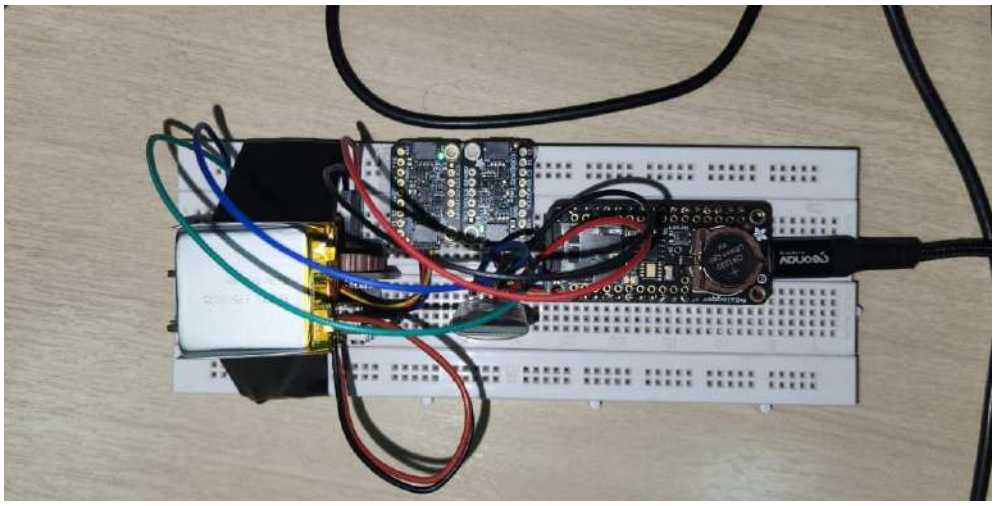


Figura 3.13: Sensores posicionados para o teste comparativo entre os modelos ISM330DHCX e LSM6DSOX

Adicionalmente, realizou-se um teste comparativo entre os dois sensores, ISM330DHCX e LSM6DSOX, percorrendo uma rota pré-definida, com ambos os sensores montados lado a lado, conforme Figura 3.13. A análise dos sinais obtidos, dispostos na Figura 3.14 e Figura 3.15, revelou que a diferença entre os sensores, para a aplicação desenvolvida, é insignificante, uma vez que eles possuem o mesmo formato temporal e a mesma assinatura espectral. Portanto, optou-se pelo modelo *Adafruit ISM330DHCX*, por possuir maior robustez em relação a altas temperaturas por um custo similar ao modelo LSM6DSOX.

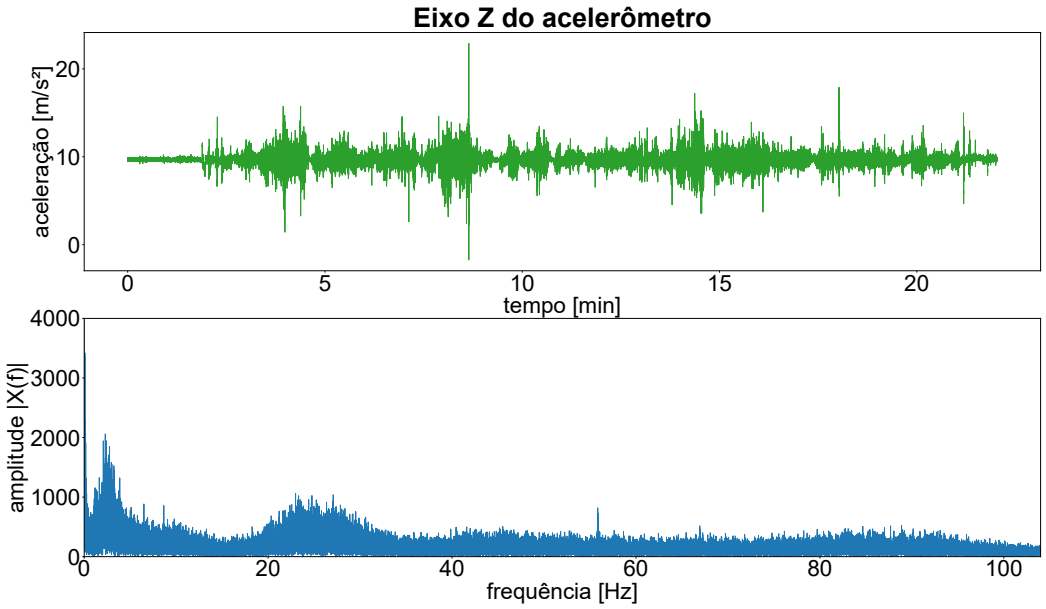


Figura 3.14: Sinal capturado para o teste comparativo, sensor ISM330DHCX

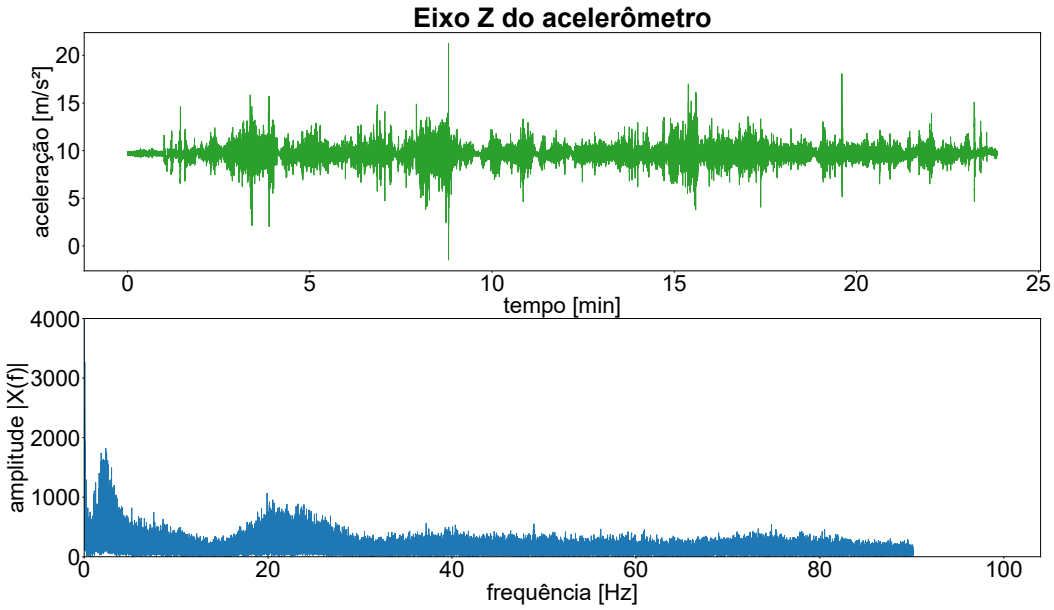


Figura 3.15: Sinal capturado para o teste comparativo, sensor LSM6DSOX

Por fim, uma vez selecionadas as variáveis do sensor (frequência de amostragem, modelo, nível de sensibilidade), foi realizada uma saída de campo para consolidar os resultados finais. Percorreu-se um trecho de 15 quilômetros, compostos majoritariamente por pavimento perfeito e desgastado, conforme classificação da CNT, e o resultado obtido está disposto na Figura 3.16.

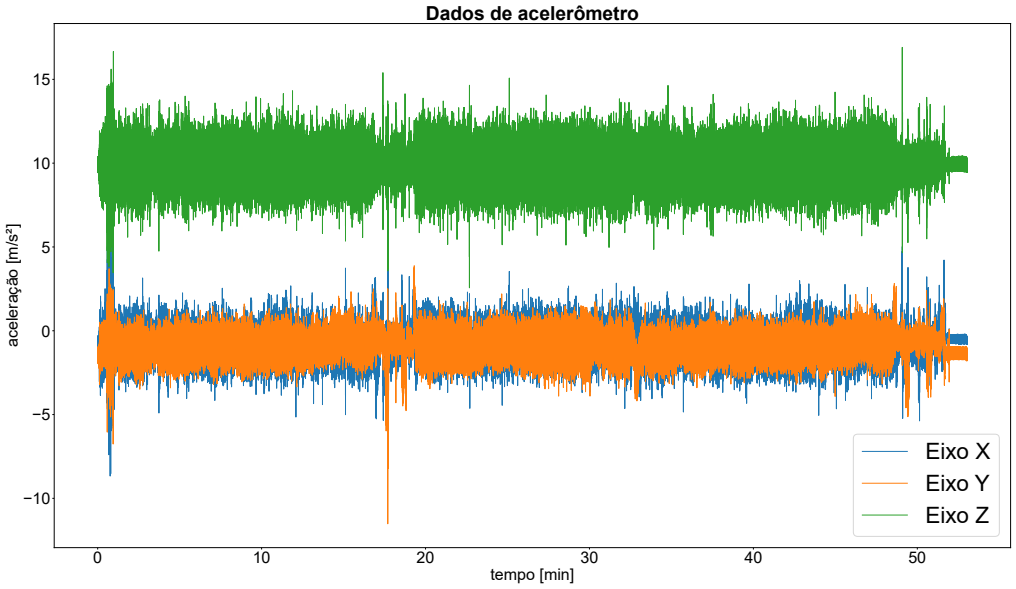


Figura 3.16: Dados de acelerômetro para a saída final de verificação do sistema embarcado

Dessarte, as configurações finais selecionadas para o sensor acelerômetro/giroscópio podem ser vistas na Tabela 3.1. O sistema embarcado com essas configurações foi utilizado para sensoriamento dos dados de vibração do pavimento, que posteriormente são usados para classificação via modelo de *machine learning*.

Elemento	Parâmetro	Valor
Acelerômetro	Frequência de amostragem	104 Hz
	Faixa de operação	±16G
Giroscópio	Frequência de amostragem	104 Hz
	Faixa de operação	±500 dps
Cartão SD	Capacidade de armazenamento	8 GB

Tabela 3.1: Configurações definidas para a implementação do sistema embarcado

4 CLASSIFICAÇÃO DE VÍDEOS DO PAVIMENTO USANDO MODELO DE APRENDIZAGEM PROFUNDA

Este capítulo dedica-se à descrição do processo de desenvolvimento do modelo de classificação de vídeos e imagens do pavimento. Objetivou-se a escrita sucinta, sem abrir mão do rigor técnico necessário, das motivações e escolhas tomadas ao longo do desenvolvimento, de forma que fique claro a natureza e o objetivo deste modelo, bem como suas limitações e pontos de melhoramento futuros.

Para além da captura dos dados de sensoriamento embarcado, um dos objetivos delineados era a classificação de vídeos/imagens do pavimento, por meio de um modelo de IA. Assim, para compreender o modelo desenvolvido, é necessário antes entender o fluxo de dados da PCR no tocante aos vídeos armazenados, posteriormente detalhando-se as escolhas tomadas para a construção do modelo, bem como qual a configuração definitiva escolhida para implementação em campo.

4.1 FLUXO DE DADOS DA PESQUISA CNT DE RODOVIAS

Como mencionado anteriormente, a cada pesquisador da PCR é atribuída uma rota pré-definida, subdivididas em unidades de pesquisa (correspondentes a um trecho específico), que por sua vez são subdivididas em unidades de coleta. Cada carro de pesquisa é equipado com duas câmeras, uma em cada ponto do veículo, e os vídeos são armazenados em dispositivos MDVR (*mobile digital video recording*), conforme mostra a Figura 4.1. Para este trabalho, utilizou-se apenas os vídeos obtidos pela câmera posicionada no capô do carro, pois ela é posicionada de forma que os vídeos capturem com mais detalhes o pavimento. Exemplos de imagens extraídas da câmera do pavimento podem ser vistas na Figura 3.1.

Figura 4.1: Carro de pesquisa, com as duas câmeras utilizadas. A câmera responsável pela captura das imagens de pavimento está destacada em vermelha, localizada no capô do veículo

4.2 CRITÉRIO DE CLASSIFICAÇÃO

Antes de detalhar o modelo de classificação em si, deve-se determinar qual é o critério utilizado para o problema de classificação em questão. Conforme mencionado anteriormente, a PCR utiliza cinco classes para classificar a qualidade do pavimento. Para o este trabalho, optou-se por limitar estas classes em apenas duas: pavimento perfeito e imperfeito; portanto, reduz-se a complexidade do problema para um problema de classificação binária. A descrição da classe pavimento perfeito permanece a mesma que aquela utilizada pela PCR, e a classe pavimento imperfeito nada mais é que a união de todas as outras definições de classes utilizadas.

Assim, uma vez recebido um vídeo de UC, com duração aproximada de um minuto, deve-se utilizar o algoritmo para classificá-lo *frame* a *frame* e, posteriormente, realizar a classificação de toda a UC com base no critério de maioria, isto é, se mais da metade dos frames do vídeo foram classificados como perfeito, classifica-se o vídeo, e portanto a UC, como perfeita.

4.3 CONSTRUÇÃO DO DATASET

O *dataset* construído para o treinamento do modelo é composto de 850 imagens de treinamento, 150 de validação e 300 de teste, para cada classe. As imagens foram obtidas dos vídeos

armazenados ao longo da PCR 2023, após classificação realizada pelos pesquisadores em campo e posteriormente pela equipe interna da CNT, garantindo maior confiabilidade na rotulação do *ground truth*.

Devido à natureza continental do território brasileiro, é de se esperar que existe uma grande variabilidade entre os tipos de pavimentos para diferentes estados da Federação. Dessa forma, para garantir que as imagens do *dataset* possuíssem representatividade em relação àquilo existente na vida real, e proporcionar um bom nível de generalização do modelo, adotou-se uma métrica para cálculo do nível de semelhança entre as imagens selecionadas para o *dataset*. Mais especificamente, utilizou-se a métrica *structural similarity index measure* (SSIM), aplicando-a entre as imagens selecionadas para o *dataset*, item a item, e, caso o valor obtido estivesse abaixo do limiar $0,9$, a imagem permaneceu na base de dados. A Figura 4.2 traz um exemplo demonstrativo do cálculo do SSIM, utilizando para isso uma imagem de base, simulando uma imagem que já estivesse presente no *dataset*, enquanto outras duas imagens são comparadas a ela e decide-se se entram ou não na base de dados.

(a) Imagem base utilizada para calcular o SSIM

(b) Imagem reprovada com SSIM $> 0,9$

(c) Imagem aprovada com SSIM $\leq 0,9$

Figura 4.2: Exemplo de cálculo da métrica SSIM

Dessa forma, prosseguiu-se com a construção do *dataset*, até atingir-se as quantidades desejadas. A seguir, na Figura 4.3, estão representadas uma imagem de exemplo da base de dados, para cada classe, dos conjuntos de treinamento, validação e teste.

(a) Imagem perfeita do *dataset* de treinamento

(b) Imagem imperfeita do *dataset* de treinamento

(c) Imagem perfeita do *dataset* de validação

(d) Imagem imperfeita do *dataset* de validação

(e) Imagem perfeita do *dataset* de teste

(f) Imagem imperfeita do *dataset* de teste

Figura 4.3: Exemplos de imagens do *dataset* de treinamento, validação e teste

Como os resultados expostos deixarão claro, o *dataset* originalmente construído não foi suficiente para garantir um desempenho satisfatório dos modelos, com algumas alterações, em termos de processamento da imagem original da pesquisa, se mostrando necessárias.

4.4 PRIMEIRO MODELO DESENVOLVIDO

Para a construção do primeiro modelo, optou-se pelo uso da rede de aprendizagem profunda ResNet50, constituída de 50 camadas convolucionais e residuais, conforme descrito na Seção

2.3.4. Na saída da rede ResNet, concatenou-se uma rede composta de duas camadas totalmente conectadas, com 256 e 2 neurônios, respectivamente.

A entrada do modelo consiste de imagens de tamanho 1920x1080 pixels, no formato obtido pelas câmeras da PCR, sem nenhuma modificação, e no mesmo formato do *dataset* original construído. A *pipeline* do modelo pode ser vista na Figura 4.4.

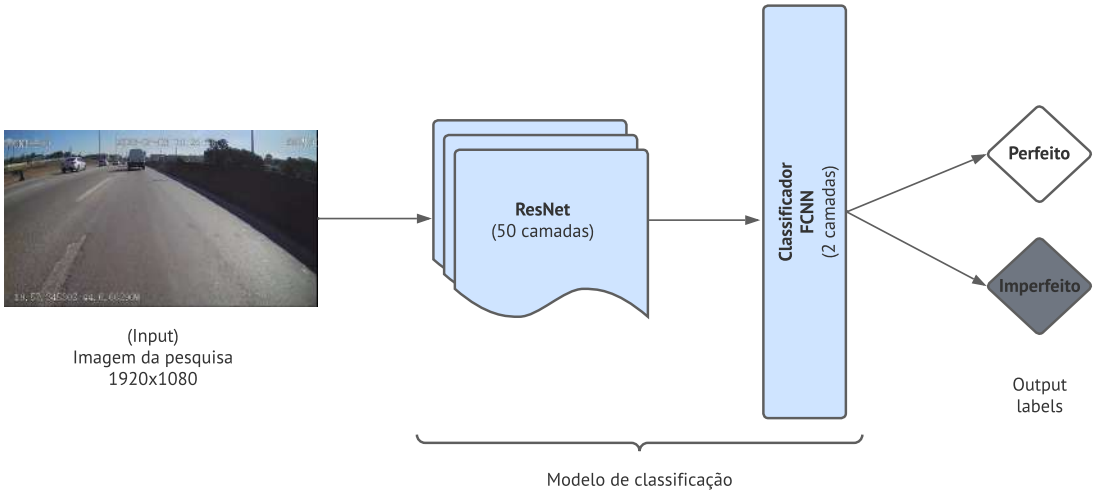
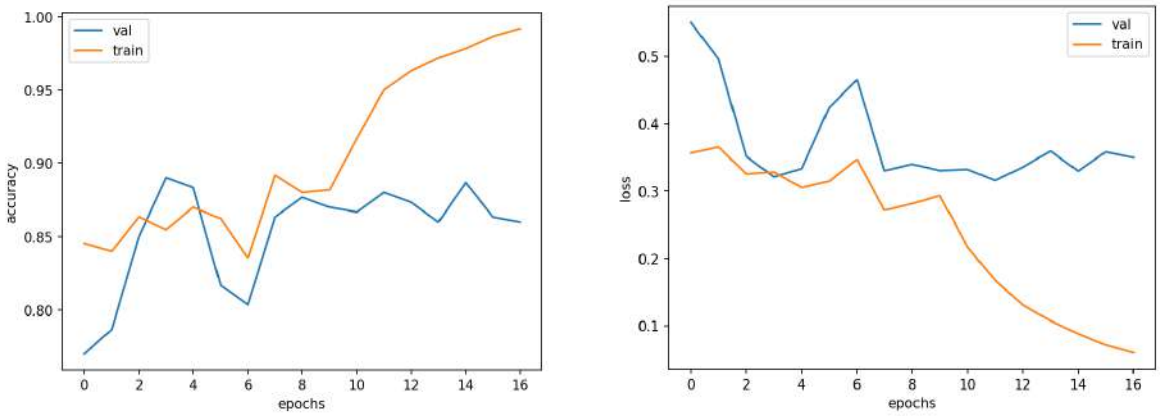


Figura 4.4: *Pipeline* do primeiro modelo de classificação

4.4.1 Resultados obtidos

O modelo foi treinado por 100 épocas, mas, por convergência do resultado (valor de erro permaneceu marginalmente inalterado por 5 épocas ou mais), encerrou-se o treinamento na 16ª época. Os resultados de precisão e erro do modelo, ao longo do treinamento e da validação, podem ser visualizados na Figura 4.5. Adicionalmente, a precisão de teste do modelo foi de $91,17\%$.



(a) Dados de precisão

(b) Dados de erro

Figura 4.5: Dados de treinamento e validação, para o primeiro modelo desenvolvido

4.4.2 Análise dos resultados

Computou-se a matriz de confusão do modelo, disposta na Figura 4.6, para permitir uma análise mais detalhada do modelo.

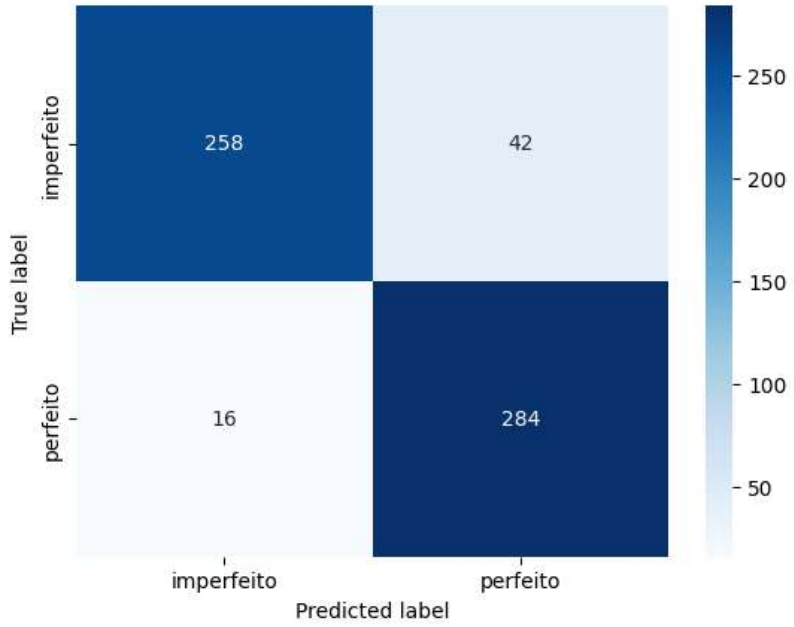


Figura 4.6: Matriz de confusão do primeiro modelo desenvolvido

O cálculo da precisão de teste ($P_{t\%}$) individualmente para cada classe, fazendo

$$P_{t_{\text{imperfeito}}\%} = \frac{258}{300} = 86,00\%\,,$$

$$P_{t_{\text{perfeito}}\%} = \frac{284}{300} = 94,67\%\,,$$

indica que o modelo está desbalanceado e desempenhando melhor para os casos perfeitos.

Analisando os exemplos de imagens utilizadas no *dataset*, na Figura 4.3, percebe-se que estas incluem significativo nível de contexto, isto é, partes da imagem que não são de interesse para o problema de classificação em questão, como cenas de paisagem, outros veículos, pórticos e defensas. Assim, lançou-se mão da hipótese de que, caso as imagens utilizadas no *dataset* fossem previamente processadas, removendo o contexto, a precisão do modelo poderia aumentar. Para testar essa hipótese, modificou-se o *dataset* original e retreinou-se o modelo.

4.5 SEGUNDO MODELO DESENVOLVIDO

Conforme mencionado anteriormente, uma das hipóteses lançadas ao analisar o desempenho do primeiro modelo era a de que o contexto presente nas imagens de treinamento estariam influenciando negativamente a precisão do modelo.

Nesse sentido, optou-se por realizar o corte espacial das imagens do *dataset* em 25% da altura, transformando as imagens de 1920x1080 pixels em 1920x810 pixels. O corte das imagens permitiu, como efeito adicional, a melhoria do tempo de execução do modelo, por trabalhar com imagens menores.

A Figura 4.7 traz um exemplo do processamento realizado. Esse processo foi repetido para todas as imagens do *dataset* de treinamento, validação e teste.

(a) Exemplo perfeito do *dataset* original, de tamanho 1920x1080

(b) Exemplo perfeito processado, com contexto removido, de tamanho 1920x810

Figura 4.7: Processamento de imagem realizado no *dataset* original

A arquitetura do modelo permaneceu a mesma, alterando-se apenas a *pipeline* de entrada do modelo, conforme Figura 4.8.

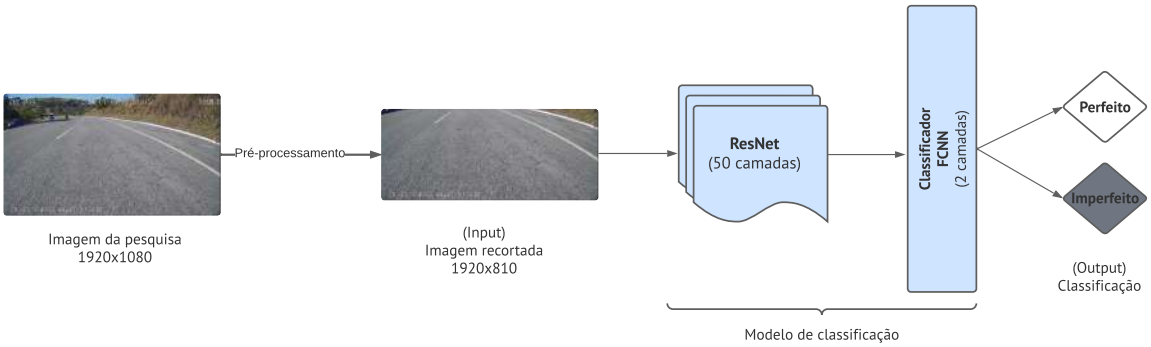
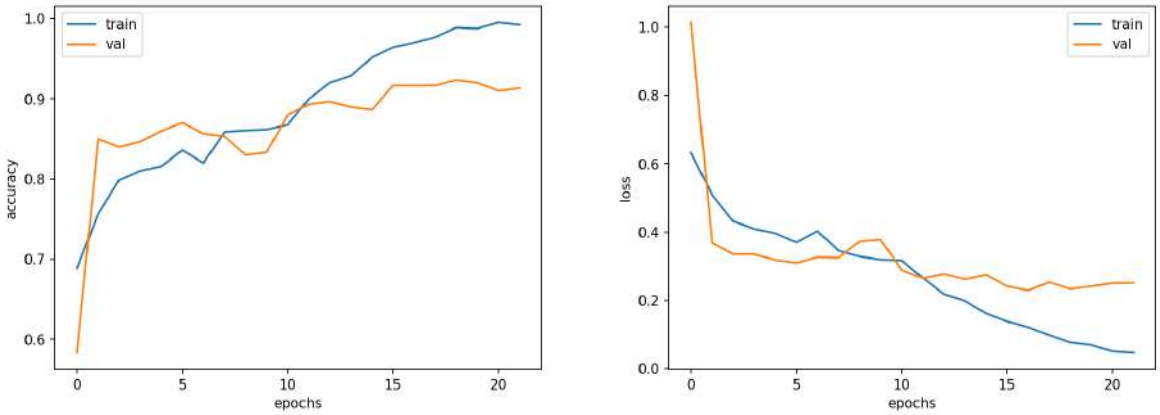


Figura 4.8: *Pipeline* do segundo modelo de classificação

4.5.1 Resultados obtidos

O modelo foi treinado, novamente, por 100 épocas, mas a convergência dos resultados encerrou o treinamento na 22ª época. Os resultados de precisão e erro de treinamento e validação estão expostos na Figura 4.9 e, além disso, a precisão de teste foi de $91,67\%$.



(a) Dados de precisão

(b) Dados de erro

Figura 4.9: Dados de treinamento e validação, para o segundo modelo desenvolvido

4.5.2 Análise dos resultados

A análise das curvas de precisão e erro demonstram que o treinamento do modelo se comportou de forma mais suave que no treinamento anterior. A matriz de confusão do modelo está disposta na Figura 4.10.

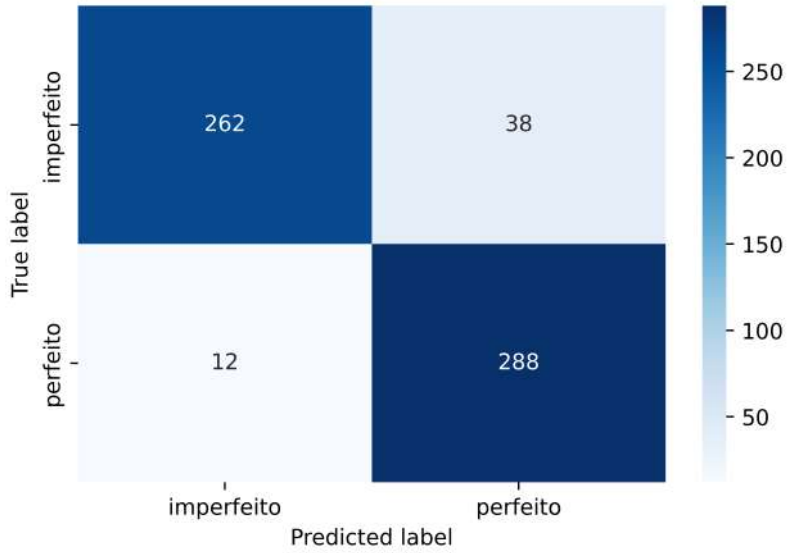


Figura 4.10: Matriz de confusão do segundo modelo desenvolvido

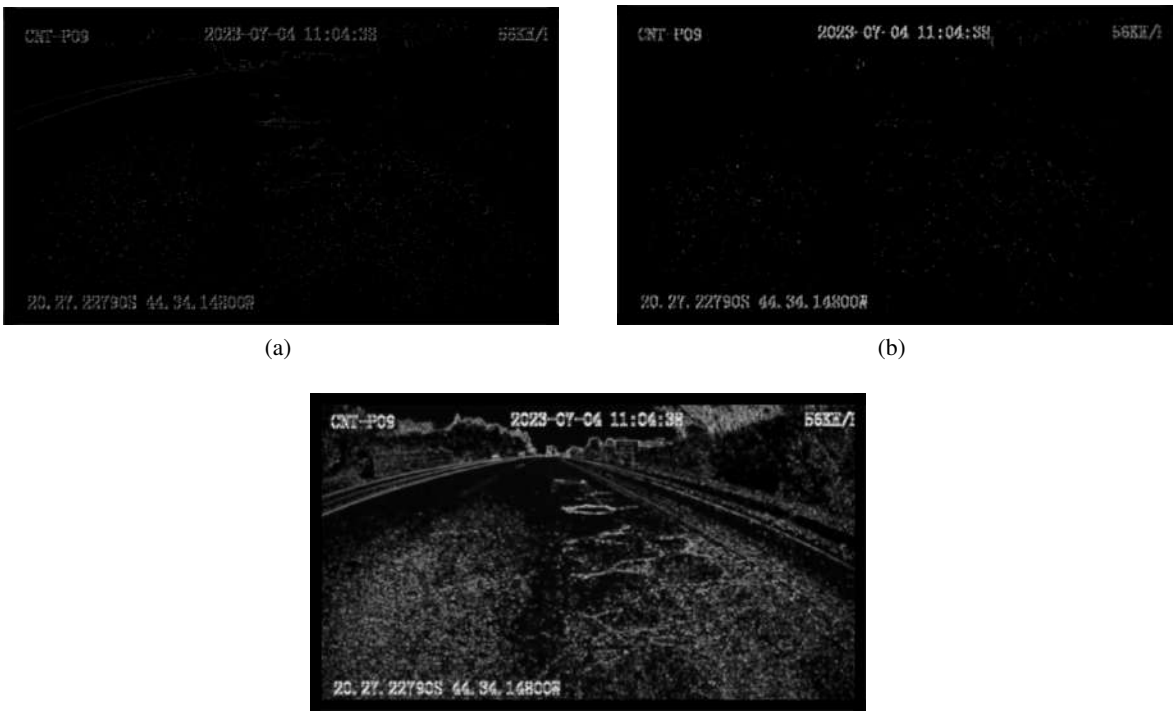
O cálculo da precisão de teste ($P_{t\%}$) individualmente para cada classe, fazendo

$$P_{t_{\text{imperfeito}}\%} = \frac{262}{300} = 87,34\%\,,$$

$$P_{t_{\text{perfeito}}\%} = \frac{288}{300} = 96,00\%\,,$$

indica que o modelo está desbalanceado e desempenhando melhor para os casos perfeitos, mas menos que no modelo anterior, confirmando a hipótese de que o contexto estaria influenciando negativamente o desempenho do modelo.

Para aprofundar a análise do modelo, selecionou-se algumas imagens para avaliar os *feature maps* produzidos por elas, com exemplos dispostos na Figura 4.11. A análise desses mapas de características revela que o modelo, por vezes, estava selecionando caracteres da tela, referentes aos dados do MDVR e do carro de pesquisa, como coordenadas, velocidade, número do carro e horário. Nesse sentido, traçou-se a hipótese de que isso estaria influenciando positivamente o desempenho do modelo, isto é, que o modelo estava aprendendo a reconhecer esses caracteres nas imagens de treinamento e, em imagens de teste com caracteres parecidos, estava classificando-os com a mesma classe.



(a) (b)

(c)

Figura 4.11: Exemplos de *feature maps* produzidos na terceira camada convolucional

De forma a testar a razoabilidade dessa hipótese, realizou-se uma nova adaptação do *dataset*, dessa vez removendo não somente 25% do topo da imagem, mas também realizando um corte na parte inferior, de forma que a imagem resultante compreendesse somente o pavimento, sem informação do contexto (removido anteriormente) e nem de caracteres (nova modificação proposta).

4.6 MODELO DE CLASSIFICAÇÃO DEFINITIVO

Para construção do modelo definitivo, modificou-se o *dataset* original com a remoção de 25% dos pixels da parte superior, bem como a remoção dos caracteres da parte inferior das imagens. Assim, as imagens utilizadas para treinamento do modelo definitivo ficaram com tamanho 1920x721 pixels. A Figura 4.12 mostra um exemplo do processamento realizado, procedimento que foi repetido para todas as imagens do *dataset* de treinamento, validação e teste.

(a) Exemplo imperfeito do *dataset* original, de tamanho 1920x1080 pixels

(b) Exemplo imperfeito processado, com caracteres inferiores e contexto removido, de tamanho 1920x721 pixels

Figura 4.12: Processamento de imagem realizado para construção do *dataset* definitivo

Além disso a isso, adicionou-se uma nova camada intermediária totalmente conectada ao final do modelo, que antes contava com duas camadas de 256 e 2 neurônios respectivamente. Assim, o modelo definitivo conta com 1000, 256 e 2 neurônios, dispostos em três camadas totalmente conectadas. A saída do modelo consiste de uma tupla com as probabilidades de cada classe, provenientes de uma função de ativação do tipo sigmóide. A Tabela 4.1 resume as características fundamentais do modelo, e a Figura 4.13 representa a *pipeline* final do modelo definitivo.

Dado	**Valor**
Número de camadas	53
Número de parâmetros	25,8 M
Taxa de inferência	48,09 $\frac{\text{inf.}}{\text{s}}$
Tamanho em disco	98,7 MB

Tabela 4.1: Dados importantes do modelo de classificação de vídeos

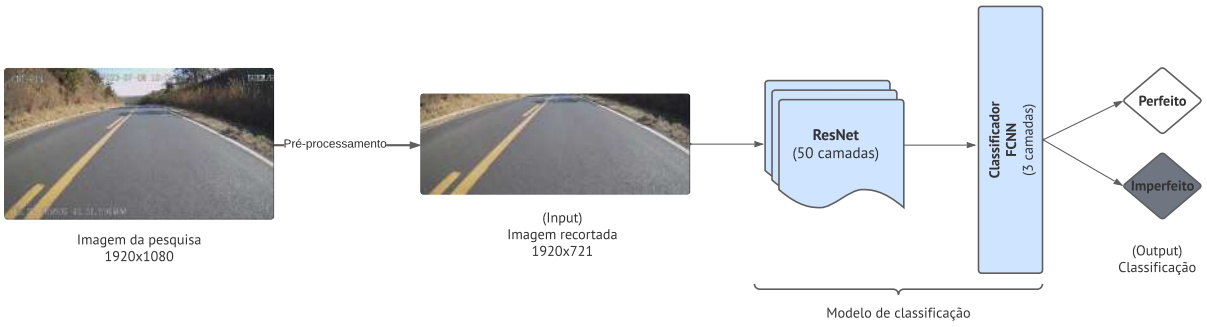


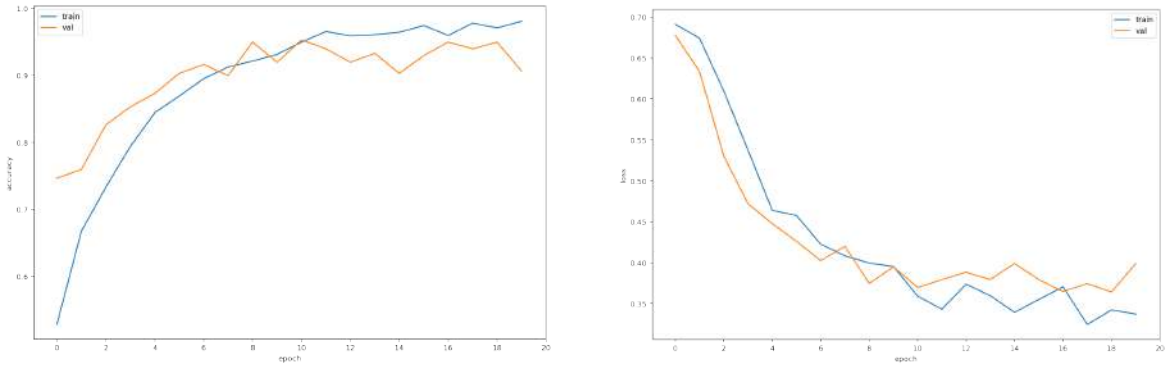
Figura 4.13: *Pipeline* do modelo definitivo

Todo o desenvolvimento do modelo deu-se utilizando a linguagem de programação Python, que oferece uma infinidade de bibliotecas para o desenvolvimento de modelos de *machine learning*. Especificamente, utilizou-se a biblioteca PyTorch, lançando mão da implementação do modelo ResNet50 disponível.

4.6.1 Resultados obtidos

O modelo foi treinado por 100 épocas, mas devido à convergência antecipada, o treinamento se encerrou na 18ª época. O *hardware* utilizado foi a placa de vídeo NVIDIA RTX A5500, o processador Intel Xeon W3-2435 e 32 GB de memória RAM, com duração do treinamento de aproximadamente trinta minutos. O modelo utilizou o otimizador SGD, com parâmetro de *momentum* igual a $0,9$.

Os dados de precisão e erro de treinamento e validação podem ser vistos na Figura 4.14 e, além disso, a precisão de teste foi de $91,67\%$.



(a) Dados de precisão

(b) Dados de erro

Figura 4.14: Dados de treinamento e validação, para o modelo definitivo

4.6.2 Análise dos resultados

Analisando as curvas de precisão e erro, conclui-se que o treinamento ocorreu de forma suave. Além disso, conforme mostra a Figura 4.15, percebe-se que o erro de classificação para o caso perfeito reduziu significativamente. O erro de classificação para o caso imperfeito aumentou, embora não esteja ruim.

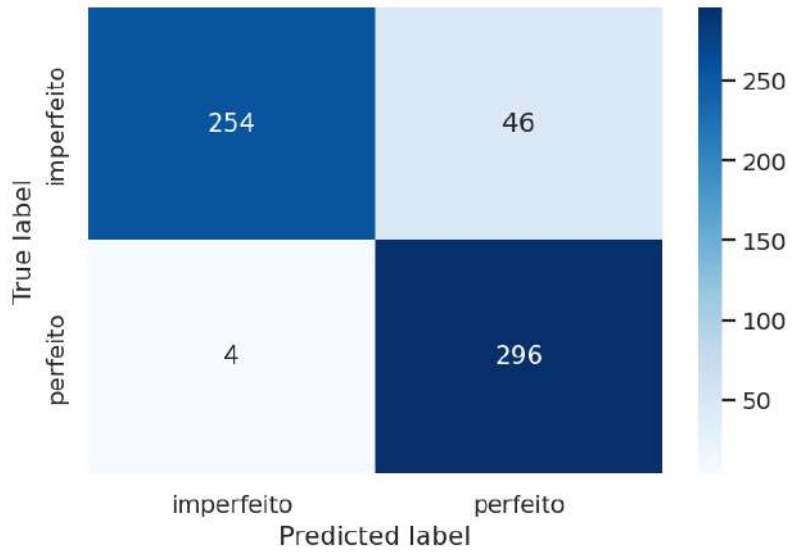


Figura 4.15: Matriz de confusão do modelo definitivo

O cálculo da precisão de teste ($P_{t\%}$) individualmente para cada classe, fazendo

$$P_{t_{\text{imperfeito}}\%} = \frac{254}{300} = 84,67\%\,,$$

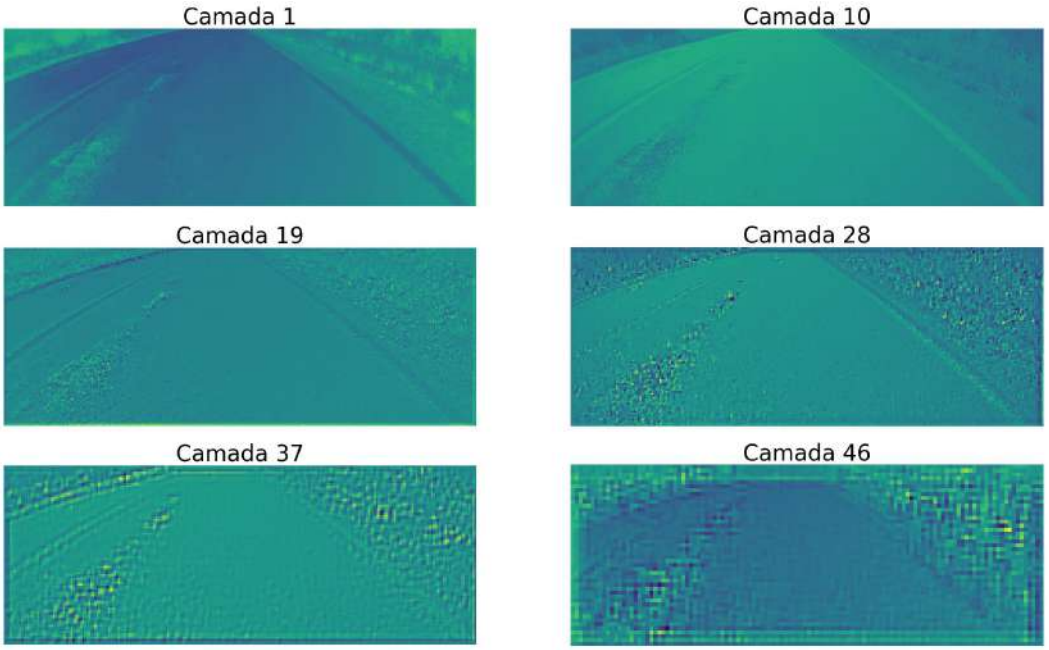
$$P_{t_{\text{perfeito}}\%} = \frac{296}{300} = 98,67\%\,,$$

indica que o modelo está desbalanceado e desempenhando melhor para os casos perfeitos. A redução de precisão para os exemplos imperfeitos pode ser justificada pela remoção dos caracteres, realizada para o treinamento do modelo definitivo, pois, conforme explicado, o modelo estava aprendendo a reconhecer os caracteres e classificando imagens com caracteres parecidos como pertencentes à mesma classe.

Em termos computacionais, utilizando o *hardware* supracitado, cada inferência realizada pelo modelo durou cerca de 0,02 segundos. Para obtenção deste dado, executou-se o *script* contido no apêndice I.1. O modelo possui cerca de 25,8 milhões de parâmetros, e ocupa um espaço em disco de 98,7 MB.

Destaca-se, ainda, que a saída do modelo, por ser proveniente de uma função de ativação do tipo sigmóide, produz uma tupla com as probabilidades da imagem de entrada ser pertencente às classes perfeito ou imperfeito e, então, define-se a classe da imagem de entrada como aquela que possui maior probabilidade entre as duas. A probabilidade correspondente à classe selecionada pelo modelo é denominada de nível de confiança ($\mathcal{N}.\mathcal{C}.$), valor este exposto para os dois exemplos selecionados nas Figuras 4.16 e 4.17.

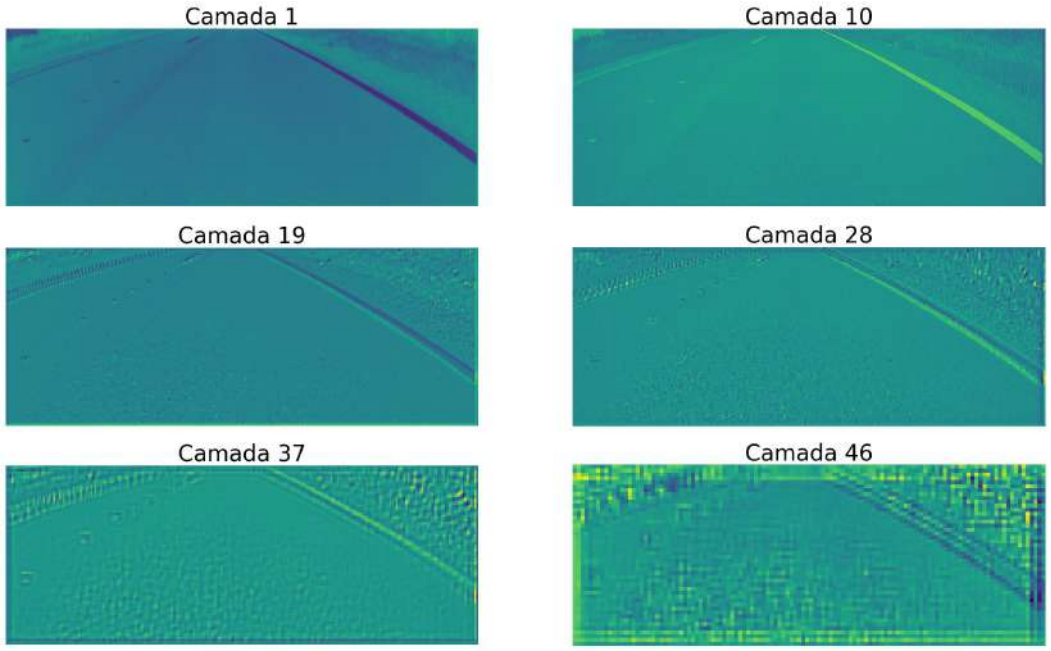
(a) Imagem de teste classificada como imperfeita, com $\mathcal{NC}. = 0,99999$



(b) Exemplos de *feature maps* gerados

Figura 4.16: Teste de *feature maps* do modelo definitivo

(a) Imagem de teste classificada como perfeita com $\mathcal{NC}. = 0,99946$



(b) Exemplos de *feature maps* gerados

Figura 4.17: Teste de *feature maps* do modelo definitivo

Por fim, a análise de *feature maps* das Figuras 4.16 e 4.17 revela que o modelo está, de fato, aprendendo a reconhecer os padrões de patologia, e não está destacando, desta vez, nenhum carácter presente na imagem.

4.7 VALIDAÇÃO CRUZADA UTILIZANDO O MÉTODO K-FOLDS

Para validar os dados obtidos pelo modelo definitivo, realizou-se a validação cruzada utilizando o método *k-folds*. Esse método consiste em unificar os *datasets* de treinamento e validação em um só, e subdividir essa nova base de dados em K pastas. Em seguida, realiza-se o treinamento do modelo K vezes, sendo que em cada uma das vezes, uma das pastas é selecionada como *dataset* de testes e as outras $K - 1$ pastas são usadas para treinamento/validação do modelo. A

Figura 4.18 representa esquematicamente como funciona o processo.

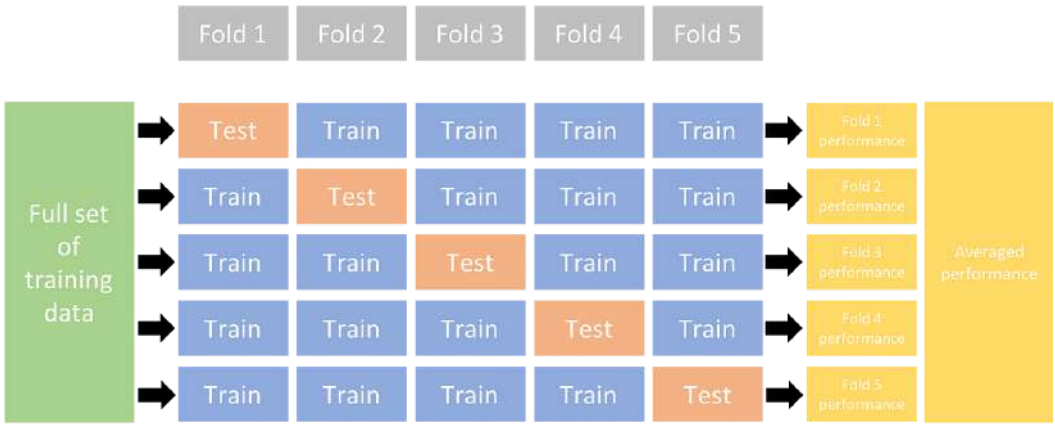
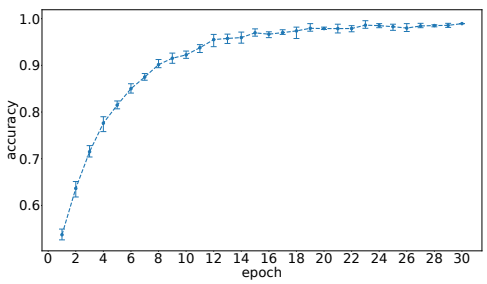
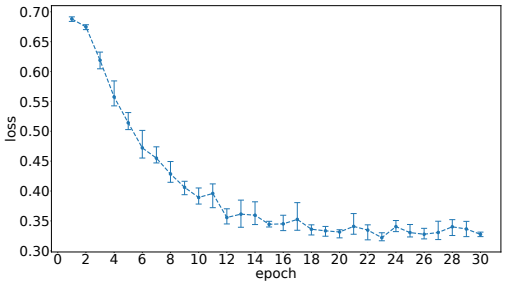


Figura 4.18: Diagrama com a explicação simplificada do método de validação cruzada *k-folds*, com $K = 5$

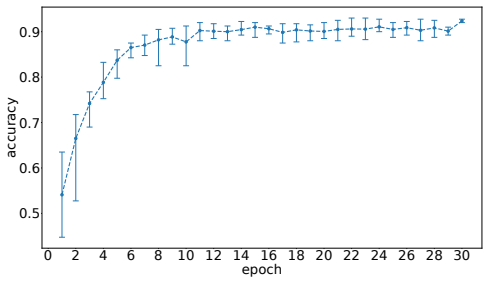
Para o modelo definitivo construído, selecionou-se $K = 5$ e aplicou-se o método treinando o modelo por 30 épocas. Os resultados obtidos podem ser vistos na Figura 4.19.



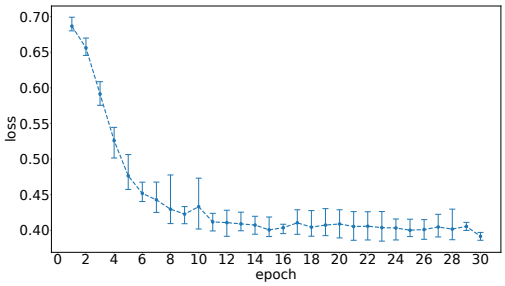
(a) Precisão de treinamento média



(b) Erro de treinamento médio



(c) Precisão de validação média



(d) Erro de validação médio

Figura 4.19: Resultados da validação cruzada usando o método *k-folds*, com $K = 5$. As barras de erro representam o intervalo de valores obtidos nas cinco iterações e a linha pontilhada é média

Observando as curvas obtidas, conclui-se que o modelo está convergindo para os resultados obtidos no primeiro treinamento, revelando que ele não possui nenhuma forma de vício de treinamento para uma configuração de *dataset* de treinamento específica. Adicionalmente, as barras de erro mostram valores bem comportados dentro de um limiar finito, corroborando com a conclusão

feita.

4.8 CLASSIFICAÇÃO DE VÍDEOS

Um vídeo pode ser entendido como uma sequência, ou empilhamento, de *frames* consecutivos onde, a cada segundo, uma certa quantidade de *frames* é exibida, dando a sensação de movimento. Nesse sentido, uma vez que a pesquisa PCR é armazenada em forma de vídeos em dispositivos MDVR, a implementação do modelo de classificação de pavimento deve ser feita em vídeos.

A lógica fundamental por trás dessa implementação é que o modelo deve selecionar um *frame* a cada certa unidade de tempo, classificá-lo, e armazenar as classificações de alguma forma. Uma vez que o vídeo correspondente a uma UC for percorrido, realizar-se-á a classificação da UC usando o critério de maioria, isto é, se metade dos *frames* selecionados for classificado como pertencente à classe $\mathcal{C}_1$, a UC será classificada como pertencente à classe $\mathcal{C}_1$, e vice-versa.

Figura 4.20: Teste da distância de captura máxima da câmera de pavimentos, onde a linha vermelha representa a distância de 8 metros

Uma análise mais aprofundada acerca da câmera utilizada para captura de vídeos do pavimento revela que a distância ideal máxima para gravação é de aproximadamente 8 a 10 metros, conforme indica a Figura 4.20. Assim, considerando que o veículo de pesquisa é instruído a percorrer a 60 km/h, calcula-se o tempo necessário para percorrer 8 metros, dada por

$$T_a = \frac{3600\ [\text{s}] \cdot 8\ [\text{m}]}{60000\ [\text{m}]} \approx 0,5\ [\text{s}]\,.$$

Nesse sentido, o modelo deve selecionar, a cada 0,5 segundos, um *frame* para ser analisado, correspondente aos 8 metros que a câmera é capaz de capturar com boa qualidade. Em outras pa-

lavras, o modelo deve ter uma *frequência de amostragem de classificação* de, aproximadamente,

$$f_{\text{classificação}} = \frac{1}{T_a} = \frac{1}{0,5} = 2 \text{ [fps]} .$$

Como o MDVR captura os vídeos a 15 fps, isso significa que a cada sete *frames* armazenados, o modelo classificará um.

A escolha conservadora por 8 metros, ao invés de 10, dá-se pelo fato de que, caso o veículo de pesquisa ultrapasse o limite superior de velocidade, haverá ainda certo grau de sobreposição entre os *frames*, e não haverá risco de perda de informação do pavimento.

4.8.1 Exemplo demonstrativo

Para elucidar o método de classificação de vídeos/UCs adotado será feito um exemplo demonstrativo, com um esquema didático disposto no diagrama da Figura 4.21.

Figura 4.21: Esquemático didático representando a classificação de um vídeo, corresponde a uma UC

Considerando um vídeo de aproximadamente um minuto, referente a uma UC aleatoriamente selecionada, percorre-se o vídeo, selecionando-se aproximadamente um *frame* a cada 0,5 segundos. A cada frame selecionado, o modelo realiza uma predição e esta predição é armazenada em um arquivo de registro, no formato *.csv* (valor separado por vírgula). Às classificações perfeitas são dadas o valor 1, e para as imperfeitas o valor 0. Ao final do vídeo, calcula-se a classe da UC rodando o *script* disposto a seguir:

```
1      prediction_score = sum(predictions_video)/len(predictions_video)
2      label = 'perfeito' if prediction_score>0.5 else 'imperfeito'
```

onde *predictions_video* é uma variável do tipo lista, e armazena os valores de predição para cada *frame* selecionado.

Esse procedimento é repetido para todas as UCs da pesquisa, dia-a-dia, e os resultados são agregados pela equipe da CNT.

5 CLASSIFICAÇÃO DE DADOS DE SENSORIAMENTO USANDO REDE NEURAL CONVOLUCIONAL

Este capítulo descreve o processo de desenvolvimento do modelo de classificação de sinais sensoriados, obtidos por meio do sistema embarcado desenvolvido. É feita, também, a investigação da natureza complementar entre os dois modelos de classificação construídos neste trabalho.

Conforme disposto na introdução deste trabalho, um dos objetivos era o desenvolvimento de um modelo de classificação capaz de receber como entrada os sinais sensoriados via *hardware* embarcado. Para cumprir com essa requisição, seguiu-se o desenvolvimento do modelo de forma análoga àquela realizada para construção do modelo de classificação de vídeos, conforme descrito minuciosamente no Capítulo 4.

As etapas fundamentais para construção do modelo são a escolha de uma arquitetura apropriada para o problema de classificação em questão, construção de um *dataset* de treinamento, validação e teste do modelo, análise dos resultados e ajuste fino. A seguir estão dispostas as etapas e escolhas feitas ao longo do desenvolvimento.

5.1 OBJETIVOS ESPECÍFICOS

A primeira vista, pode parecer ambíguo dois modelos de classificação para um único problema: a classificação do pavimento percorrido pelo carro de pesquisa da PCR. Entretanto, a natureza do modelo de classificação de sinais é intrinsecamente diferente daquela do modelo de classificação de vídeos, pois um modelo trabalha com sinais de vibração diretamente sensoriados do veículo que se desloca pelo pavimento, enquanto o outro baseia-se na análise de imagens capturadas por câmeras externas.

Ao optar por dois modelos de classificação com naturezas distintas, lançou-se mão da hipótese de que, a depender do tipo de patologia de pavimento analisada, um dos modelos se comportará melhor que o outro e, combinando as duas classificações, obter-se-á um resultado mais confiável.

Similarmente ao modelo de classificação de vídeos, optou-se por desenvolver um modelo de classificação binário, entre pavimento do tipo perfeito e do tipo imperfeito.

5.2 CONSTRUÇÃO DO DATASET

O *dataset* foi construído utilizando dados sensoriados pelo sistema embarcado desenvolvido no Capítulo 3. Foi realizada uma saída de campo com o objetivo de coleta de dados suficientes para o desenvolvimento da base de dados, percorrendo trechos perfeitos e trechos que compõe a

classe imperfeita, com defeitos do tipo desgaste, trinca em malha, remendos e buracos.

Para a seleção dos trechos do sinal correspondentes a amostras perfeitas e imperfeitas, utilizou-se os vídeos do trecho percorrido e selecionou-se manualmente as amostras. A Figura 5.1 mostra um exemplo de separação manual feita para um trecho de um minuto, com os recortes manuais de amostras sendo feitos posteriormente.

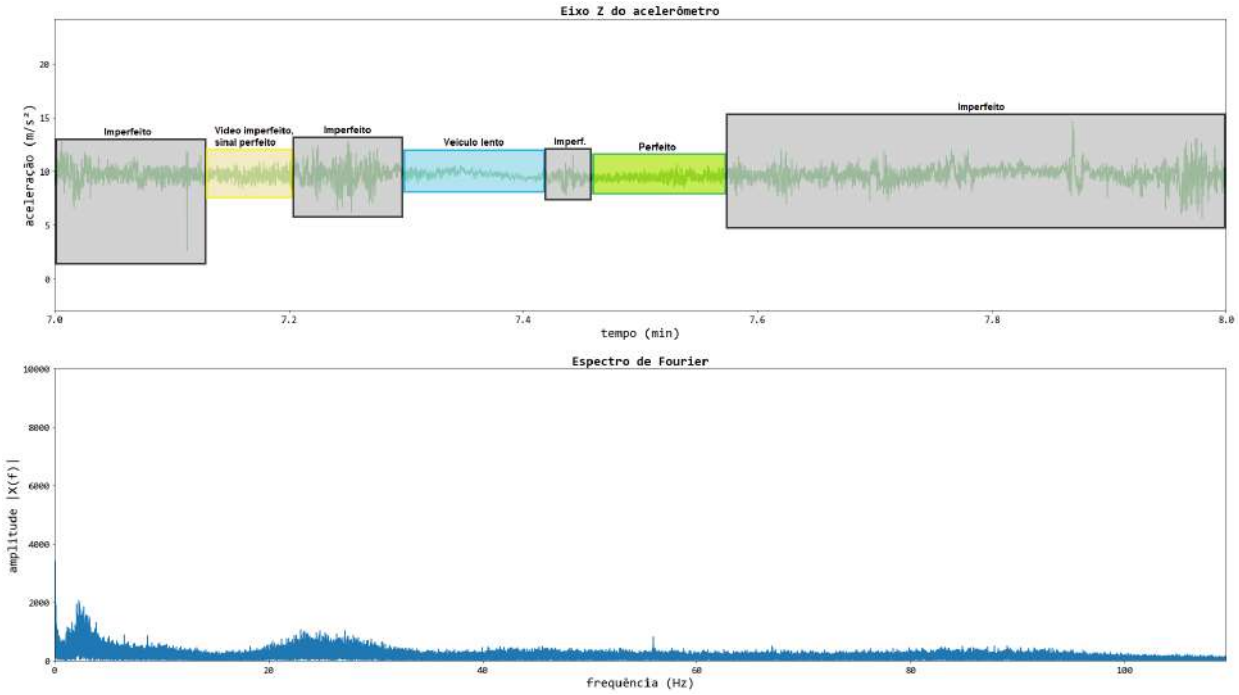


Figura 5.1: Exemplo de trecho rotulado manualmente, de onde extraiu-se as amostras perfeitas e imperfeitas

5.2.1 Cálculo do janelamento do sinal sensoriado

Como descrito no capítulo de construção do SE, os sensores operam a uma taxa de amostragem de 104 Hz, ou seja, a cada segundo são armazenadas 104 amostras do sinal, para cada eixo dos sensores (eixo X, Y e Z no acelerômetro e giroscópio). Para determinar o tamanho das janelas do *dataset*, isto é, da quantidade de amostras que serão inseridas como entrada do modelo, utilizou-se alguns critérios. Inicialmente, calculou-se a quantidade de amostras referente à distância do entre-eixos do veículo de pesquisa, quando este trafega a uma velocidade de 60 km/h. A distância entre-eixos do veículo utilizado para coleta dos dados era de aproximadamente 2,57 metros, então, calculou-se o tamanho do janelamento como

$$t_{\text{entre-eixos}} = \frac{2,57 \cdot 3600}{60000} = 0,1542 \text{ [s]},$$

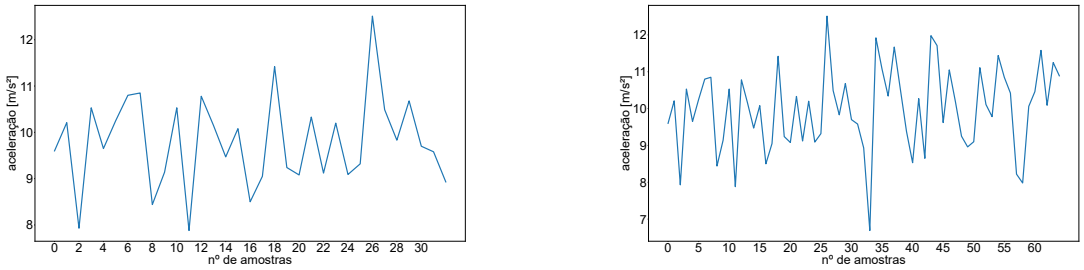
onde $t_{\text{entre-eixos}}$ é o tempo necessário para percorrer a distância do entre-eixos de 2,57 metros, quando se trafega a 60 km/h, e, então,

$$n_{\text{janelamento}} = 104 \cdot 0,1542 \approx 17 \text{ amostras},$$

onde $n_{\text{janelamento}}$ é o número de amostras na janela.

Esse valor possui um grande inconveniente em termos de custo computacional, por não ser um valor potência de dois, o que pode traduzir em maior tempo de treinamento e execução das predições do modelo, bem como dificuldades para convergência dos resultados. Portanto, selecionou-se a janela de 32 amostras para teste, por ser o valor potência de dois mais próximo acima de 17.

Além disso, como deseja-se eventualmente sincronizar os dois modelos para tomar uma decisão conjunta, convém que a distância de pavimento que o modelo de vídeos analisa seja a mesma que o modelo de sinais classifica. Assim, como foi utilizado uma distância de 8 metros para o modelo de vídeos, realiza-se o mesmo cálculo disposto acima, de forma análoga, para essa nova distância. O resultado obtido é de uma janela de aproximadamente 50 amostras, o que implica em uma janela selecionada para testes de 64 amostras, a potência de dois mais próxima de 50. Nesse sentido, de forma resumida, dois tamanhos de janela foram selecionados para teste do modelo, uma de 32 amostras e outra de 64 amostras, com exemplos dispostos na Figura 5.2.



(a) Exemplo de sinal janelado com 32 amostras

(b) Exemplo de sinal janelado com 64 amostras

Figura 5.2: Sinais de exemplo dos janelamentos selecionados para teste

5.3 ARQUITETURA DO MODELO

Para a seleção da arquitetura do modelo, levou-se em consideração a natureza dos sinais sensoriados pelo sistema embarcado. São utilizados dois sensores, com três graus de liberdade cada, correspondente aos eixos X, Y e Z do acelerômetro e giroscópio. Por opção de desenvolvimento, optou-se por fornecer todos os eixos dos dois sensores como entrada do modelo, ou seja, a entrada do modelo é um tensor do tipo $\{b, e, n\}$, onde b representa o *batch size*, e representa o número de graus de liberdade dos dois sensores somados e n representa a quantidade de amostras da janela escolhida.

Nesse sentido, selecionou-se uma rede do tipo convolucional clássica, composta de duas ca-

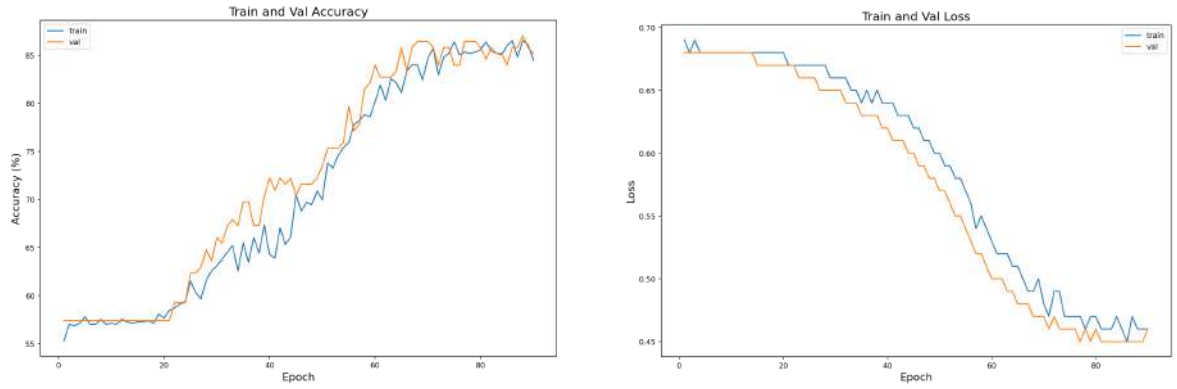
madas, concatenada a uma rede totalmente conectada também de duas camadas. Além disso, após cada camada convolucional, adicionou-se uma camada do tipo *dropout*, com o objetivo de randomicamente desativar alguns neurônios, o que melhora o desempenho de treinamento da rede ao reduzir o risco de *overfitting*, como demonstra (Srivastava et al. 2014). A entrada do modelo depende do tamanho da janela selecionada, com testes sendo realizados para definir entre 32 e 64 amostras.

5.4 RESULTADOS OBTIDOS

Realizou-se o treinamento do modelo para os dois tipos de janelamento selecionados, 32 e 64 amostras. Utilizou-se *hardware* placa de vídeo NVIDIA RTX A5500, processador Intel Xeon W3-2435 e 32 GB de memória RAM. Todo o desenvolvimento do modelo deu-se usando a linguagem de programação Python e a biblioteca PyTorch. Os resultados obtidos para cada caso estão dispostos a seguir.

5.4.1 Modelo com janelamento de 32 amostras

Selecionou-se o janelamento composto por 32 amostras e sem sobreposição entre as janelas, e realizou-se treinamento por 90 épocas com *dataset* composto de 34608 amostras, sendo 19831 perfeitas (57,3%) e 14777 imperfeitas (42,7%). Obtiveram-se as curvas de precisão e erro de treinamento e validação, dispostas na Figura 5.3.



(a) Dados de precisão

(b) Dados de erro

Figura 5.3: Dados de treinamento e validação, para o modelo de classificação de sinais sensoriados, com janelamento de 32 amostras

Adicionalmente, gerou-se a matriz de confusão do modelo, disposta na Figura 5.4. A precisão de treinamento obtida foi de aproximadamente 82,82%.

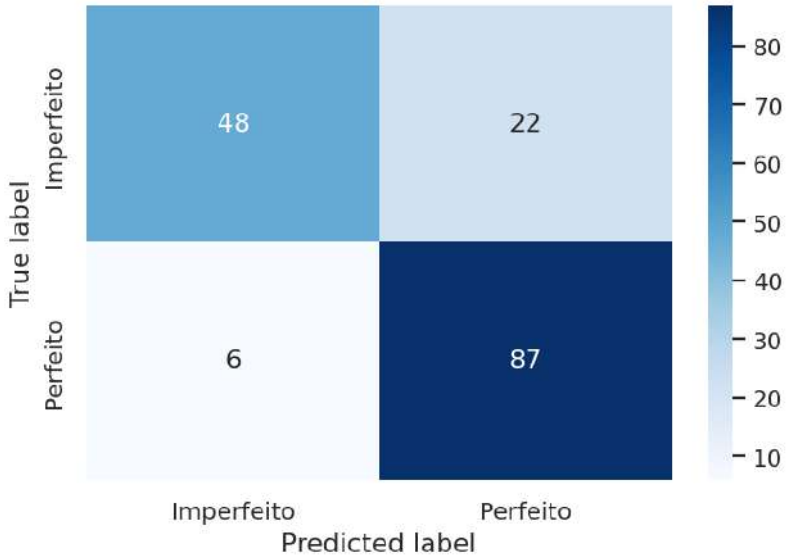
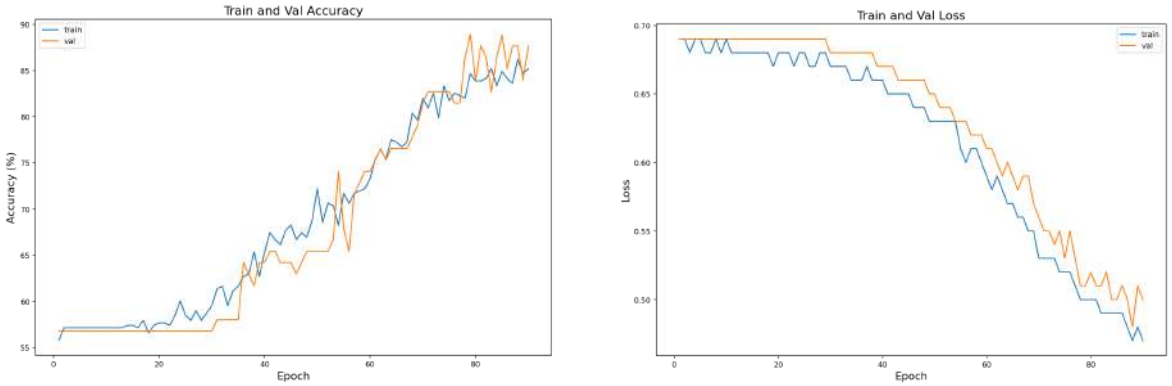


Figura 5.4: Matriz de confusão do modelo de classificação de sinais sensoriados, com janelamento de 32 amostras

5.4.2 Modelo com janelamento de 64 amostras

Selecionou-se o janelamento composto por 64 amostras e sem sobreposição entre as janelas, e realizou-se treinamento por 90 épocas com *dataset* composto de 34608 amostras, sendo 19831 perfeitas (57,3%) e 14777 imperfeitas (42,7%). Obteve-se as curvas de precisão e erro de treinamento e validação, dispostas na Figura 5.5.



(a) Dados de precisão

(b) Dados de erro

Figura 5.5: Dados de treinamento e validação, para o modelo de classificação de sinais sensoriados, com janelamento de 64 amostras

Adicionalmente, gerou-se a matriz de confusão do modelo, disposta na Figura 5.6. A precisão de treinamento obtida foi de aproximadamente 87,65%.

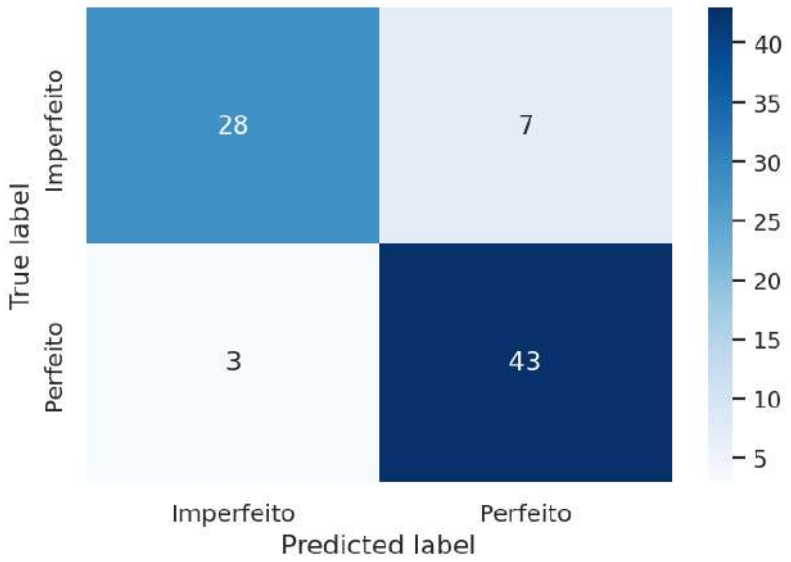


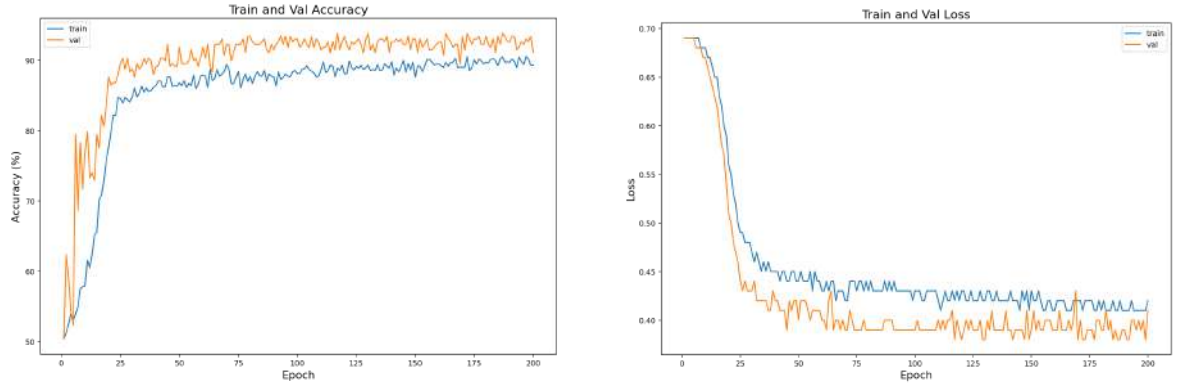
Figura 5.6: Matriz de confusão do modelo de classificação de sinais sensoriados, com janelamento de 64 amostras

5.4.3 Modelo com janelamento de 64 amostras e refinamento manual do dataset

Por fim, selecionou-se o janelamento composto por 64 amostras e com sobreposição de 32 amostras. Isso significa que, a cada exemplo de treinamento, a entrada é composta de 64 amostras, onde 32 dessas amostras também foram selecionadas pelo exemplo de treinamento anterior.

Para além disso, realizou-se refinamento manual do *dataset*, percorrendo novamente o vídeo de referência do dia de coleta dos dados e garantindo que os exemplos das classes perfeito/imperfeito eram realmente corretos. O *dataset* refinado totalizou 55144 amostras, sendo 27332 perfeitas (49,6%) e 27812 imperfeitas (50,4%).

O modelo foi treinado por 200 épocas, obtendo-se as curvas de precisão e erro de treinamento e validação dispostas na Figura 5.7.



(a) Dados de precisão

(b) Dados de erro

Figura 5.7: Dados de treinamento e validação, para o modelo de classificação de sinais sensoriados, com janelamento de 64 amostras, sobreposição de 32 amostras e refinamento manual do *dataset*

Adicionalmente, gerou-se a matriz de confusão do modelo, disposta na Figura 5.8. A precisão de treinamento obtida foi de aproximadamente 89,96%.

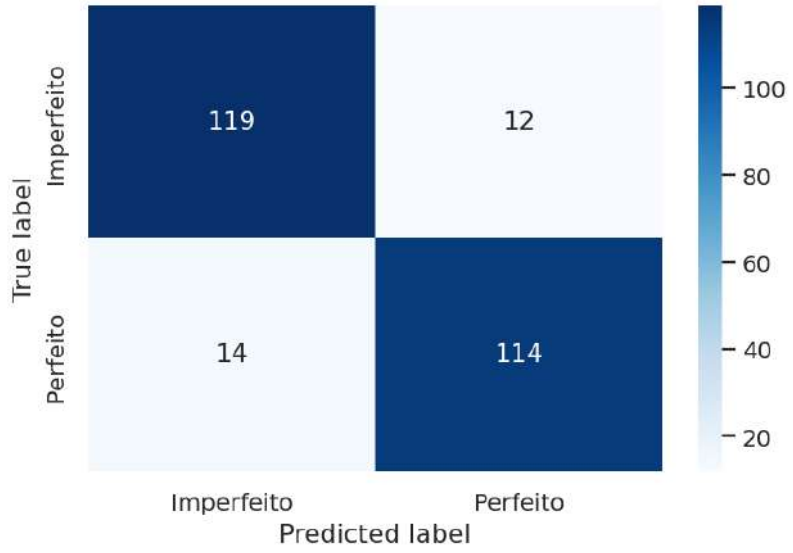


Figura 5.8: Matriz de confusão do modelo de classificação de sinais sensoriados, com janelamento de 64 amostras, sobreposição de 32 amostras e refinamento manual do *dataset*

5.5 ANÁLISE DOS RESULTADOS

A análise dos dados de treinamento realizado com janela de 32 amostras mostrou um resultado insatisfatório para a classe imperfeita, conforme pode ser visto na respectiva matriz de confusão da Figura 5.4. Por sua vez, o teste com janelamento de 64 amostras revelou uma significativa melhora, o que revela que esse tamanho de janela é o ideal para o modelo.

Entretanto, ao analisar as curvas de precisão e erro de treinamento e validação da Figura 5.5, observa-se que o treinamento por 90 épocas não foi suficiente para convergir para um resultado, com o modelo apresentando ainda espaço para redução do erro e melhoria da precisão, e portanto optou-se por realizar treinamento por 200 épocas. Adicionalmente, foi realizado um refinamento manual do *dataset*, ao observar-se que alguns exemplos selecionados para cada classe estavam mal classificados.

Essas modificações permitiram um resultado significativamente superior, como demonstra as curvas da Figura 5.7, bem como a matriz de confusão da Figura 5.8 e a precisão de treinamento de 89,96%.

Esse modelo, treinado por 200 épocas com base de dados refinada manualmente, janelamento de 64 amostras e sobreposição de 32 amostras foi selecionado como modelo definitivo para aplicação em campo.

5.6 APLICAÇÃO EM CAMPO

A lógica de aplicação do modelo de sensores em campo segue uma ideia parecida com aquela do modelo de classificação de vídeos, isto é, os dados de sensores são separados a nível de UC, a partir dos dados de marcação realizados pela pesquisadora, e a UC é classificada "janela a janela". Após isso, classifica-se toda UC com base no critério de maioria.

A principal diferença entre a aplicação prática dos modelos é o fluxo de dados, uma vez que os dados de sensores são armazenados em cartão SD, e portanto basta enviá-los para a máquina de processamento e separá-los em UCs, ao final de um dia de pesquisa. Já os dados de vídeos, precisam primeiro ser extraídos do MDVR via *software* proprietário.

6 CONCLUSÃO

Neste capítulo serão feitos os comentários finais acerca dos resultados obtidos com as tecnologias desenvolvidas. Ademais, será feita um vislumbre de perspectivas futuras, partindo das conclusões obtidas neste trabalho, de forma a contribuir ainda mais para a Pesquisa CNT de Rodovias.

6.1 COMENTÁRIOS FINAIS

Ao longo deste projeto foi possível desenvolver três frentes de trabalho multidisciplinares, a citar:

- O desenvolvimento de um sistema embarcado, capaz de realizar o sensoriamento de vibrações mecânicas do pavimento;
- O desenvolvimento de um modelo de *machine learning* capaz de realizar classificações do pavimento, baseado em imagens e vídeos da PCR;
- O desenvolvimento de um modelo de *machine learning* capaz de realizar classificações do pavimento, baseado em sinais sensoriados de vibração.

As soluções propostas aqui atendem de forma parcial ou completamente às necessidades elencadas pela equipe da CNT, dispostas na introdução deste trabalho, e constituem um robusto ponto de partida para novas tecnologias e soluções para os anos a seguir.

Em relação ao sistema embarcado, foi possível construir um sistema funcional de sensoriamento de vibrações, capaz de detectar seis graus de liberdade com capacidade de amostragem de até 833 Hz, muito acima do necessário para sinais desse tipo. Como demonstraram os exaustivos testes realizados e expostos no Capítulo 3, o sistema também mostrou-se resiliente a falhas de *software*, a elementos ambientais como altas temperaturas, e a quantidade de dados, com o correto dimensionamento da capacidade de armazenamento via cartão SD provando-se capaz de armazenar todos os dados gerados em toda a Pesquisa CNT de Rodovias.

No tocante ao modelo de classificação de imagens e vídeos, atingiu-se um alto grau de precisão, tanto em termos de classificação de *frames* quanto em classificação de vídeos (unidades de coleta). Diversos testes de arquiteturas foram realizados para seleção da configuração ideal, com o resultado final sendo caracterizado por uma arquitetura robusta e testada em diversas aplicações, a citar o modelo ResNet50, concatenado com três camadas totalmente conectadas. Os resultados obtidos foram ainda validados utilizando o método *k-folds*, o que indica boa capacidade de generalização do modelo.

Ao que concerne o modelo de classificação de sinais sensoriados, obteve-se, também, alto grau de precisão, em termos de classificação de janelas selecionadas. Testes relativos à classificação

de UCs não foram realizados, pois o desenvolvimento necessário para isso dependia de ações internas de pessoal da CNT, em relação à sincronia do tablet de pesquisa com o modelo, e não houve tempo hábil na agenda da CNT para realizar isso. Entretanto, os resultados obtidos parecem promissores e constituem uma base sólida para continuação dos desenvolvimentos.

6.2 PERSPECTIVAS FUTURAS

Os desenvolvimentos realizados permitem a construção de projetos futuros, utilizando este trabalho como base. Assim, sugere-se, a seguir, algumas melhorias e pontos de desenvolvimento que podem partir das conclusões aqui obtidas.

Um dos pontos principais é a unificação do modelo de classificação de imagens com o modelo de classificação de sinais. A lógica por trás dessa ideia, conforme exposto no Capítulo 5, é a capacidade complementar da natureza desses dois modelos, permitindo com que decisões mais acertadas a cerca da classificação do pavimento sejam realizadas. Encontra-se, na literatura, algumas estratégias acerca do fusionamento de dados, com as mais típicas concentrando-se em concatenar as duas decisões dos dois modelos em um esquema de votação, ou, alternativamente, unificar os dois dados em um modelo único. O trabalho feito por Nadeem et al. (2021) fornece uma abordagem válida para o problema de fusionamento de dados e modelos.

Outro ponto interessante a se considerar para continuação do trabalho aqui desenvolvido é a expansão dos modelos de classificação binário para modelos de classificação multi-classes. Conforme exposto no Capítulo 1, a PCR classifica os tipos de pavimento em cinco classes distintas, e, portanto, é lógico concluir que um modelo ainda mais robusto de classificação partiria para o problema de múltiplas classes.

Pode-se, ainda, expandir o modelo de imagens e vídeos para não somente classificação, como também detecção ou até mesmo segmentação dos tipos de patologias, o que geraria informações inéditas para a CNT, e alavancaria em muito a robustez da pesquisa. Algumas sugestões metodológicas podem ser verificas nos trabalhos de Li et al. (2020), Cao, Liu e He (2020) e Li, Zhao e Li (2019).

Adicionalmente, em relação ao sistema embarcado, pode-se complementar o que já foi desenvolvido com a adição de um dispositivo do tipo *global positioning system* (GPS), responsável por retornar as coordenadas a cada instante em que um novo dado é salvo. Não obstante, pode-se ainda implementar a comunicação com o tablet de pesquisa, integrando de forma definitiva os dados coletados pelo pesquisador em campo, com os dados obtidos via sensoriamento.

Conclui-se, por fim, que foram atingidos todos os objetivos elencados na introdução deste trabalho. Nesse sentido, os resultados obtidos mostram o desenvolvimento de tecnologias robustas e que têm desempenho excelente para os propósito para o qual foram criadas. Ademais, os desenvolvimentos realizados permitem a continuidade de pesquisas e desenvolvimentos na área, o que eventualmente pode contribuir para a Pesquisa CNT de Rodovias não somente em termos

de redução da carga de trabalho dos pesquisadores, mas também como fonte adicional de dados, contribuindo positivamente para o engrandecimento da pesquisa.

REFERÊNCIAS BIBLIOGRÁFICAS

Aboah e Adu-Gyamfi 2020 ABOAH, A.; ADU-GYAMFI, Y. Smartphone-based pavement roughness estimation using deep learning with entity embedding. *Advances in Data Science and Adaptive Analysis*, World Scientific, v. 12, n. 03n04, p. 2050007, 2020.

Cao, Liu e He 2020 CAO, W.; LIU, Q.; HE, Z. Review of pavement defect detection methods. *Ieee Access*, IEEE, v. 8, p. 14531–14544, 2020.

Chun, Yamane e Tsuzuki 2021 CHUN, P.-j.; YAMANE, T.; TSUZUKI, Y. Automatic detection of cracks in asphalt pavement using deep learning to overcome weaknesses in images and gis visualization. *Applied Sciences*, MDPI, v. 11, n. 3, p. 892, 2021.

CNT 2022 CNT. Pesquisa CNT de Rodovias 2022: relatório gerencial. *Brasília: CNT*, 2022.

Colavite e Konishi 2015 COLAVITE, A. S.; KONISHI, F. A matriz do transporte no brasil: uma análise comparativa para a competitividade. *Simpósio de Excelência em Gestão e Tecnologia*, v. 12, p. 28, 2015.

Cooley e Tukey 1965 COOLEY, J. W.; TUKEY, J. W. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, v. 19, n. 90, p. 297–301, 1965.

Gonzalez 2009 GONZALEZ, R. C. *Digital image processing*. [S.l.]: Pearson education india, 2009.

Haykin 2009 HAYKIN, S. *Neural networks and learning machines, 3/E*. [S.l.]: Pearson Education India, 2009.

He et al. 2016 HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778.

Heath 2002 HEATH, S. *Embedded systems design*. [S.l.]: Elsevier, 2002.

Lathi e Green 2005 LATHI, B. P.; GREEN, R. A. *Linear systems and signals*. [S.l.]: Oxford University Press New York, 2005. v. 2.

LeCun, Bengio et al. 1995 LECUN, Y.; BENGIO, Y. et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, Citeseer, v. 3361, n. 10, p. 1995, 1995.

LeCun et al. 1995 LECUN, Y.; JACKEL, L.; BOTTOU, L.; BRUNOT, A.; CORTES, C.; DENKER, J.; DRUCKER, H.; GUYON, I.; MULLER, U.; SACKINGER, E. et al. Comparison of learning algorithms for handwritten digit recognition. In: PERTH, AUSTRALIA. *International conference on artificial neural networks*. [S.l.], 1995. v. 60, n. 1, p. 53–60.

Lee e Seshia 2017 LEE, E. A.; SESHIA, S. A. *Introduction to embedded systems: A cyber-physical systems approach*. [S.l.]: MIT press, 2017. v. 2.

Li et al. 2020 LI, G.; WAN, J.; HE, S.; LIU, Q.; MA, B. Semi-supervised semantic segmentation using adversarial learning for pavement crack detection. *IEEE Access*, IEEE, v. 8, p. 51446–51459, 2020.

Li, Zhao e Li 2019 LI, J.; ZHAO, X.; LI, H. Method for detecting road pavement damage based on deep learning. In: SPIE. *Health Monitoring of Structural and Biological Systems XIII*. [S.l.], 2019. v. 10972, p. 517–526.

Marques e Almeida 2004 MARQUES, M. d. C. d. C.; ALMEIDA, J. J. M. d. Auditoria no sector público: um instrumento para a melhoria da gestão pública. *Revista Contabilidade & Finanças*, SciELO Brasil, v. 15, p. 84–95, 2004.

McCulloch e Pitts 1943 MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943.

Nadeem et al. 2021 NADEEM, M. W.; GOH, H. G.; PONNUSAMY, V.; ANDONOVIC, I.; KHAN, M. A.; HUSSAIN, M. A fusion-based machine learning approach for the prediction of the onset of diabetes. In: MDPI. *Healthcare*. [S.l.], 2021. v. 9, n. 10, p. 1393.

Panwar et al. 2017 PANWAR, M.; DYUTHI, S. R.; PRAKASH, K. C.; BISWAS, D.; ACHARYYA, A.; MAHARATNA, K.; GAUTAM, A.; NAIK, G. R. Cnn based approach for activity recognition using a wrist-worn accelerometer. In: IEEE. *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. [S.l.], 2017. p. 2438–2441.

Rosenblatt 1958 ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, American Psychological Association, v. 65, n. 6, p. 386–408, 1958.

Sholevar, Golroo e Esfahani 2022 SHOLEVAR, N.; GOLROO, A.; ESFAHANI, S. R. Machine learning techniques for pavement condition evaluation. *Automation in Construction*, Elsevier, v. 136, p. 104190, 2022.

Shtayat et al. 2023 SHTAYAT, A.; MORIDPOUR, S.; BEST, B.; ABUHASSAN, M. Using supervised machine learning algorithms in pavement degradation monitoring. *International Journal of Transportation Science and Technology*, Elsevier, v. 12, n. 2, p. 628–639, 2023.

Silva, Spatti e Flauzino 2016 SILVA, I. N. d.; SPATTI, D. H.; FLAUZINO, R. A. *Redes neurais artificiais para engenharia e ciências aplicadas*. [S.l.]: Artliber, 2016.

Srivastava et al. 2014 SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014.

Zhang et al. 2021 ZHANG, A.; LIPTON, Z. C.; LI, M.; SMOLA, A. J. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021.

APÊNDICES

I.1 CÓDIGO FONTE EM PYTHON PARA CÁLCULO DA TAXA DE INFERÊNCIA DO MODELO DEFINITIVO

```
dummy_input = torch.randn(8, 3, 1920, 721, dtype=torch.float).to(device)

repetitions=100
total_time = 0
with torch.no_grad():
    for rep in range(repetitions):
        starter, ender = torch.cuda.Event(enable_timing=True),
                         torch.cuda.Event(enable_timing=True)
        starter.record()
        _ = model(dummy_input)
        ender.record()
        torch.cuda.synchronize()
        curr_time = starter.elapsed_time(ender)/1000
        total_time += curr_time
Throughput =   (repetitions*8)/total_time
print('Final Throughput:',Throughput)
```