

ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA
MESTRADO EM ENGENHARIA ELÉTRICA

DOUGLAS BORBA

**APLICAÇÃO DE REDES NEURAIS ARTIFICIAIS PARA CLASSIFICAR
PADRÕES DE RUÍDO ELETROMAGNÉTICO CONDUZIDO NAS LINHAS DE
ALIMENTAÇÃO DE CIRCUITOS INTEGRADOS**

Porto Alegre
2022

PÓS-GRADUAÇÃO - *STRICTO SENSU*



Pontifícia Universidade Católica
do Rio Grande do Sul

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA**

DOUGLAS BORBA

**Aplicação de redes neurais artificiais para classificar padrões de ruído
eletromagnético conduzido nas linhas de alimentação de circuitos
integrados**

Porto Alegre

2022

DOUGLAS BORBA

**Aplicação de redes neurais artificiais para classificar padrões de ruído
eletromagnético conduzido nas linhas de alimentação de circuitos
integrados**

Dissertação apresentada como requisito para a obtenção do grau de Mestre pelo Programa de Pós-Graduação em Engenharia Elétrica da Escola Politécnica da Pontifícia Universidade Católica do Rio Grande do Sul.

Área de concentração: Sinais, Sistemas e Tecnologia da Informação.

Linha de Pesquisa: Sistemas de Computação, Controle e Automação.

Orientador: Prof. Dr. Fabian Luis Vargas

Porto Alegre

2022

Ficha Catalográfica

B726a Borba, Douglas

Aplicação de redes neurais artificiais para classificar padrões de ruído conduzido nas linhas de alimentação de circuitos integrados / Douglas Borba. – 2022.

105 p.

Dissertação (Mestrado) – Programa de Pós-Graduação em Engenharia Elétrica, PUCRS.

Orientador: Prof. Dr. Fabian Luis Vargas.

1. Ruído conduzido. 2. Interferência eletromagnética. 3. Rede neural artificial. 4. Sistema embarcado robusto. 5. TensorFlow. I. Vargas, Fabian Luis. II. Título.

**Aplicação de redes neurais artificiais para classificar padrões de ruído
eletromagnético conduzido nas linhas de alimentação de circuitos
integrados**

CANDIDATO: DOUGLAS BORBA

Esta Dissertação de Mestrado foi julgada para obtenção do título de MESTRE EM ENGENHARIA ELÉTRICA e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Pontifícia Universidade Católica do Rio Grande do Sul.

DR. FABIAN LUIS VARGAS - ORIENTADOR

BANCA EXAMINADORA

DR. TIAGO ROBERTO BALEN - PGMICRO - UFRGS

DR. JARBAS ARYEL NUNES DA SILVEIRA - PPGETI - UFC

DR. CÉSAR AUGUSTO MISSIO MARCON - PPGCC - PUCRS

Este trabalho é dedicado à minha esposa que esteve ao meu lado em todos os momentos me apoiando, torcendo e vibrando por minhas conquistas. Também dedico aos meus pais que abriram mão de muita coisa para poder me dar uma educação de qualidade. Dedico este trabalho a todos que torcem pela minha felicidade e sucesso.

AGRADECIMENTOS

Ao Prof. Dr. Fabian Luis Vargas pelas orientações e auxílio durante a execução deste trabalho.

Aos meus pais, Vanderlei e Janete por toda a garra, e todo amor dado a mim, contribuindo com minha educação, tudo que sou e tenho hoje dedico a eles. Amo vocês!

A minha esposa Caroline, pelo incentivo, apoio, carinho, palavras de motivação, compreensão e por todas as vezes que orou por mim dizendo que daria tudo certo. Muito obrigado, te amo!

Por fim, mas não menos importante, ao Prof. Dr. Juliano Benfica pela oportunidade de ter ingressado neste mestrado, contribuindo com o tema, com orientações e com os constantes “Toca ficha”.

A sabedoria suprema é ter sonhos bastante grandes
para não se perderem de vista enquanto
os perseguimos.

(FAULKNER, William, 1929)

RESUMO

Com o crescente uso de sistemas embarcados em nosso dia a dia e o aumento do nível de ruído eletromagnético no ambiente em que esses sistemas estão expostos, a necessidade de uma operação confiável é primordial. Quando os componentes vitais desses sistemas ficam expostos a uma grande quantidade de ruídos, que podem ser tanto conduzidos como radiados, esses ruídos podem acarretar em falhas graves e incontornáveis, por isso torna-se imprescindível a análise e testes confiáveis em cima destes componentes. Nesse cenário, este trabalho apresenta um estudo sobre o uso de técnicas de aprendizado de máquina (redes neurais artificiais) para realizar a identificação e classificação em campo de diferentes tipos de ruído eletromagnético conduzido nas linhas de alimentação de circuito integrados (CIs), de acordo com um conjunto específico das normas IEC. Este trabalho detalha o desenvolvimento de uma metodologia a qual as formas de onda dos fenômenos contidos nas normas são simuladas através de software devido ao tempo disposto para um trabalho de mestrado acrescido do viés de pandemia e a dificuldade de acesso aos laboratórios por conta deste cenário, e também pelo risco envolvendo o tempo e complexidade para obtenção dessas formas de onda em hardware utilizando um microprocessador, por exemplo. Resultados experimentais obtidos nesta metodologia, sugerem que a abordagem proposta é muito eficaz para atingir o objetivo de identificar os tipos de ruídos eletromagnéticos conduzidos. Dessa forma, a metodologia desenvolvida possibilita a inserção da inteligência artificial no contexto de testes, permitindo aos desenvolvedores de sistemas e circuitos integrados uma nova abordagem para avaliar a susceptibilidade à EMI conduzido, possibilitando o desenvolvimento de novas técnicas para o aumento da confiabilidade e robustez dos projetos.

Palavras chave: *Ruído conduzido, Interferência eletromagnética, Rede neural artificial, Sistema embarcado robusto, Anaconda Navigator, TensorFlow.*

ABSTRACT

With the increasing use of embedded systems in our daily lives and the increasing level of electromagnetic noise in the environment in which these systems are exposed, the need for reliable operation is paramount. When the vital components of these systems are exposed to a large amount of noise, which can be both conducted and radiated, these noises can lead to serious and unavoidable failures, so it is essential to analyze and reliably test these components. In this scenario, this work presents a study on the use of machine learning techniques (artificial neural networks) to carry out the identification and classification in the field of different types of electromagnetic noise conducted in the power lines of integrated circuits (ICs), according to with a specific set of IEC standards. This work details the development of a methodology in which the waveforms of the phenomena contained in the standards are simulated through software due to the time available for a master's work plus the pandemic bias and the difficulty of accessing the laboratories due to this scenario, and also by the risk involving the time and complexity to obtain these waveforms in hardware using a microprocessor, for example. Experimental results obtained in this methodology suggest that the proposed approach is very effective to achieve the objective of identifying the types of electromagnetic noise conducted. In this way, the methodology developed allows the insertion of artificial intelligence in the context of tests, allowing the developers of systems and integrated circuits a new approach to assess the susceptibility to conducted EMI, enabling the development of new techniques to increase the reliability and robustness of the projects.

Keywords: *Conducted noise, Electromagnetic interference, Machine learning, Deep neural network, Robust embedded system, Anaconda Navigator, TensorFlow.*

LISTA DE ILUSTRAÇÕES

FIGURA 1 - ESTRUTURA DE UM NEURÔNIO REAL.	23
FIGURA 2 - MODELO DE NEURÔNIO ARTIFICIAL DE MCCULLOCH E PITTS (1943).....	25
FIGURA 3 - EXEMPLO DE FUNÇÕES DE ATIVAÇÃO TRADICIONAIS.	26
FIGURA 4 - FUNÇÃO SIGMÓIDE E TANGENTE HIPERBÓLICA.	28
FIGURA 5 - FUNÇÃO RELU.	29
FIGURA 6 - RNA FEEDFORWARD COM CAMADA ÚNICA.	30
FIGURA 7 - MLP COM DUAS CAMADAS OCULTAS.	31
FIGURA 8 - SUPERFÍCIE DE ERRO DE UMA REDE MLP E A TRAJETÓRIA DO GRADIENTE DESCENDENTE.	34
FIGURA 9 - DIAGRAMA DE DEPENDABILIDADE.	35
FIGURA 10 - MODELO DOS 3 UNIVERSOS: FALHA, ERRO E DEFEITO.	38
FIGURA 11 - PROBLEMA BÁSICO DE EMC 40	40
FIGURA 12 - RELAÇÃO CUSTO <i>VERSUS</i> TÉCNICAS UTILIZADAS EM UM PROJETO COM EMC.....	41
FIGURA 13 - AMBIENTE ELETROMAGNÉTICO.	42
FIGURA 14 - EMI RADIADA.	43
FIGURA 15 - EMI CONDUZIDA.	44
FIGURA 16 - EXEMPLO DE FORMA DE ONDE DE UMA VARIAÇÃO DE TENSÃO DO TIPO SURTO.	45
FIGURA 17 - MECÂNISMO ESD.	45
FIGURA 18 - ILUSTRAÇÃO DE QUEDA E INTERRUPTÃO NA TENSÃO.	46
FIGURA 19 - EXEMPLO DE INTEGRIDADE DE SINAIS.	47
FIGURA 20 - FORMA DE ONDA PROPOSTA PELA IEC 61000-4-2.....	50
FIGURA 21 - FORMAS DE ONDA DE TRANSIENTES ELÉTRICOS RÁPIDOS 51	51
FIGURA 22 - FORMA DE ONDA DE TENSÃO DE CIRCUITO ABERTO IEC 61000-4-5.....	53
FIGURA 23 - FORMA DE ONDA DE CORRENTE DE CURTO CIRCUITO IEC 61000-4-5.....	54
FIGURA 24 - FORMA DE ONDA DA CORRENTE PARA TESTE ESD DE TENSÃO DE 2kV.....	61
FIGURA 25 - FORMA DE ONDA DA CORRENTE PARA TESTE ESD DE TENSÃO DE 4kV.....	61
FIGURA 26 - FORMA DE ONDA DA CORRENTE PARA TESTE ESD DE TENSÃO DE 6kV.....	62
FIGURA 27 - FORMA DE ONDA DA CORRENTE PARA TESTE ESD DE TENSÃO DE 8kV.....	62
FIGURA 28 - FORMAS DE ONDA PARA TESTE DE EFT COM PICOS DE 0,25kV.....	63
FIGURA 29 - FORMAS DE ONDA PARA TESTE DE EFT COM PICOS DE 0,5kV.....	63
FIGURA 30 - FORMAS DE ONDA PARA TESTE DE EFT COM PICOS DE 1kV.....	64
FIGURA 31 - FORMAS DE ONDA PARA TESTE DE EFT COM PICOS DE 2kV.....	64
FIGURA 32 - FORMA DE ONDA PARA TESTE TENSÃO DE CIRCUITO ABERTO DA IEC 61000-4-5.....	65
FIGURA 33 - FORMA DE ONDA PARA TESTE CORRENTE DE CURTO CIRCUITO DA IEC 61000-4-5.....	66
FIGURA 34 - EXEMPLOS SIMULAÇÕES GERADAS PARA TESTES DESCRITOS NA IEC 61000-4-5.....	67
FIGURA 35 - FORMA DE ONDA PARA TESTE CORRENTE DE CURTO CIRCUITO DA IEC 61000-4-5 COM RUÍDO INSERIDO 67	67
FIGURA 36 - VARIÁVEIS CRIADAS PARA REPRESENTAÇÃO DOS VETORES E SUAS RESPECTIVAS CLASSES.	69

FIGURA 37 - ARQUIVO CSV EXPORTADO COM A BASE DE DADOS	69
FIGURA 38 - FLUXO DE DESENVOLVIMENTO.....	70
FIGURA 39 - INTERFACE DO ANACONDA NAVIGATOR.....	71
FIGURA 40 - VALOR DE SAÍDA DA RNA COM TRANSFORMAÇÃO	74

LISTA DE TABELAS

TABELA 1 - CAUSAS USUAIS DE DEFEITOS EM SISTEMAS DE COMPUTAÇÃO.	38
TABELA 2 – NÍVEIS DE TESTE DA IEC 61000-4-2.	50
TABELA 3 - NÍVEIS DE TESTE DA NORMA IEC 61000-4-4.	52
TABELA 4 - NÍVEIS DE TESTE DA NORMA IEC 61000-4-5 PARA CURTO CIRCUITO.	53
TABELA 5 - NÍVEIS DE TENSÃO RECOMENDADOS PARA TESTE DE QUEDA DE TENSÃO.	55
TABELA 6 - NÍVEIS DE TENSÃO RECOMENDADOS PARA TESTE DE PEQUENA INTERRUPÇÃO DE TENSÃO.	55
TABELA 7 - NÍVEIS DE TENSÃO RECOMENDADOS PARA TESTE DE VARIAÇÃO DE TENSÃO.	56
TABELA 8 - PARÂMETROS DA FORMA DE ONDA DA CORRENTE DE ESD.	60
TABELA 9 - NORMAS E SUAS REPRESENTAÇÕES NA BASE DE DADOS	68
TABELA 10 - RESULTADAS NA PRIMEIRA FASE DE TESTES.	79
TABELA 11 - RESULTADOS NA SEGUNDA FASE DE TESTES.	80
TABELA 12 - RESULTADOS NA TERCEIRA FASE DE TESTES.	80
TABELA 13 - RESULTADOS NA QUARTA FASE DE TESTES.	81

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
CI	Circuitos Integrados
CMOS	Complementary Metal Oxide Semiconductor
CSV	Comma Separated Values
DC	Direct Current
EFT	Electrical Fast Transient
EMC	Electromagnetic Compatibility
EME	Electromagnetic Environment
EMI	Electromagnetic Interference
ESD	Electrostatic Susceptibility
I/O	Input/Output
IEC	International Electrotechnical Commission
MATLAB	Matrix Laboratory
MLP	Multilayer Perceptron
MTTF	Mean Time to Failures
RAM	Random Access Memory
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RF	Radio Frequênci
RNA	Rede Neural Artificial
SOC	System-on-a-Chip
TANH	Tangente Hiperbólica
TF	Tolerância a Falhas

SUMÁRIO

SUMÁRIO	15
I FUNDAMENTAÇÃO TEÓRICA	18
1 INTRODUÇÃO	19
1.1 OBJETIVOS	20
1.1.1 <i>Objetivos Específicos</i>	20
1.1.2 <i>Escopo do trabalho</i>	21
1.2 APRESENTAÇÃO DOS CAPÍTULOS	22
2 REDES NEURAIS ARTIFICIAIS	23
2.1 REDE NEURAL BIOLÓGICA.....	23
2.2 REDE NEURAL ARTIFICIAL	24
2.3 FUNÇÕES DE ATIVAÇÃO	26
2.3.1 <i>Função de ativação logística</i>	27
2.3.2 <i>Função de ativação Softmax</i>	28
2.3.3 <i>Função de ativação Linear Retificada (ReLU)</i>	29
2.4 ARQUITETURA DA REDE NEURAL ARTIFICIAL.....	30
2.4.1 <i>Rede neural artificial alimentada a diante com camada simples</i>	30
2.4.2 <i>Rede neural artificial com múltiplas camadas</i>	31
2.5 PROCESSOS DE APRENDIZAGEM	32
2.6 ALGORITMO DE APRENDIZAGEM POR CORREÇÃO DE ERRO.....	33
2.6.1 <i>Gradiente Descendente</i>	34
3 CONCEITOS BÁSICOS DE TOLERÂNCIA A FALHAS	35
3.1 INTRODUÇÃO	35
3.2 ATRIBUTOS	36
3.3 RECURSOS	36
3.3 AMEAÇAS	37
4 COMPATIBILIDADE ELETROMAGNÉTICA	39
4.1 INTRODUÇÃO	39
4.2 AMBIENTE ELETROMAGNÉTICO	41
4.3 INTERFERÊNCIA ELETROMAGNÉTICA (EMI)	42

4.4 EMI RADIADA E EMI CONDUZIDA	43
4.4.1 Tipos de EMI Conduzida	44
4.5 EFEITOS DA INTERFERÊNCIA ELETROMAGNÉTICA NA ELETRÔNICA	47
4.6 NORMAS E REGULAMENTAÇÕES	48
4.6.1 IEC 61000.....	49
4.6.2 IEC 61000-4-2	49
4.6.3 IEC 61000-4-4	51
4.6.4 IEC 61000-4-5	52
4.6.5 IEC 61000-4-29	54
II METODOLOGIA	57
5 SIMULAÇÃO DAS FORMAS DE ONDA E BASE DE DADOS	58
5.1 INTRODUÇÃO	58
5.2 FERRAMENTA COMPUTACIONAL PARA SIMULAÇÃO DAS FORMAS DE ONDA	58
5.3 IMPLEMENTAÇÃO DO ALGORITMO NO MATLAB	59
5.3.1 Simulação da forma de onda da IEC 61000-4-2	59
5.3.2 Simulação da forma de onda da IEC 61000-4-4	63
5.3.3 Simulação da forma de onda da IEC 61000-4-5	64
5.3.4 Simulação da forma de onda da IEC 61000-4-29	66
5.3.5 Inserção de ruído nas formas de onda simuladas	67
5.4 CRIAÇÃO DA BASE DE DADOS	68
6 CRIAÇÃO DA RNA E TREINAMENTO	70
6.1 INTRODUÇÃO	70
6.2 AMBIENTE DE DESENVOLVIMENTO	71
6.3 FERRAMENTAS UTILIZADAS PARA CRIAÇÃO E TREINAMENTO DA RNA	72
6.3.1 Pandas	72
6.3.2 Scikit-Learn	72
6.3.3 NumPy	72
6.3.4 TensorFlow	73
6.4 ARQUITETURA DA RNA	73
6.5 FUNÇÕES E PARÂMETROS UTILIZADOS NA RNA	74
6.6 TREINAMENTO E PREVISÃO DA RNA	75
III RESULTADOS E CONCLUSÕES	77

7 RESULTADOS	78
7.1 EVOLUÇÃO DA BASE DE DADOS E RELAÇÃO COM RESULTADOS.....	78
7.2 RESULTADOS OBTIDOS A CADA FASE DE TESTES	78
7.2.1 <i>Resultados da primeira fase de testes</i>	79
7.2.2 <i>Resultados da segunda fase de testes</i>	79
7.2.3 <i>Resultados da terceira fase de testes</i>	80
7.2.4 <i>Resultados da quarta fase de testes</i>	81
8 CONCLUSÃO E TRABALHOS FUTUROS.....	82
8.1 PROPOSTAS PARA TRABALHOS FUTUROS	83
REFERÊNCIAS	85
APÊNDICE A – CÓDIGO PRINCIPAL MATLAB.....	93
APÊNDICE B – FUNÇÕES CRIADAS PARA GERAR FORMAS DE ONDA - MATLAB.....	100
APÊNDICE C – ALGORITMO PARA CRIAÇÃO DA RNA – JUPYTER NOTEBOOK	104

Parte I

Fundamentação Teórica

1 INTRODUÇÃO

Com a crescente evolução da tecnologia, podemos dizer que hoje estamos vivendo uma nova revolução industrial. Cada revolução industrial teve seu marco. No século 18, a primeira revolução teve como marco a introdução das máquinas a vapor, no século 19, a segunda revolução industrial teve como marco o início da produção em massa com linhas a base de petróleo e eletricidade. Já no século 20, a terceira revolução industrial surgiu com o marco das linhas de produção automatizadas utilizando computadores (XU; DAVID; KIM, 2018). Nesta revolução a tecnologia de semicondutores viu o nascimento do transistor e dos circuitos integrados desenvolvidos e adotados em um ritmo relativamente lento. Em contrapartida, a quarta revolução industrial que estamos vivendo atualmente, está progredindo a uma taxa exponencial criando soluções totalmente novas e inovadoras (GRIFFITHS; OOI, 2018). Finalmente, nesta revolução que vivemos: a internet das coisas (ou IoT, do termo em inglês, Internet of Things), a computação em nuvem e os carros autônomos são exemplos presentes nos dias de hoje os quais possuem dispositivos que na maioria das vezes não podem tolerar nenhum tipo de falha, pois poderia acarretar em um acidente crítico, neste contexto podendo ser fatal.

Com a evolução da tecnologia os sistemas eletrônicos estão cada vez mais robustos e complexos e por sua vez, menores em sua construção, providos de maiores velocidades de operação e menor consumo de energia. A miniaturização dos circuitos integrados que utilizam tecnologias CMOS (em inglês, *Complementary Metal Oxide Semiconductor*) desde a década de 60, quando o químico Gordon Moore apresentou o trabalho (RASIT O. TOPALOGLU, 2015). Por consequência da diminuição do tamanho dos circuitos e também da tensão de alimentação, as velocidades de operação tiveram um aumento, consequentemente melhorando o desempenho dos microcontroladores (PRADHAN, 1996), mas ao mesmo tempo, aumentando a sua sensibilidade ao ruído externo, aumentando a frequência que estes podem falhar.

Além da tecnologia utilizada nos circuitos estar mudando, também os ambientes os quais eles operam passam por alterações, tornando-se mais ruidosos devido ao aumento de dispositivos conectados e diferentes tecnologias estarem sendo utilizadas em conjunto, o que causa preocupação em relação confiabilidade, desempenho e segurança (BENFICA, 2007). O uso dos microcontroladores/microprocessadores está cada vez mais presente em sistemas críticos, como por exemplo, aplicações espaciais, biomédicas, militares e industriais (BENFICA, 2015). Por tanto, a segurança destes sistemas se torna primordial e para garantir essa segurança, existem normas que certificam e qualificam estes dispositivos e é de

fundamental importância que estes circuitos operem dentro de suas especificações garantindo a segurança dos sistemas mesmo operando em ambientes ruidosos e expostos a interferências constantes. É papel dos projetistas utilizarem técnicas de prevenção e mitigação de falhas ainda na fase de projeto dos sistemas e mesmo com a importância de testes em sistemas críticos, ainda é difícil de encontrar metodologias e plataformas de testes para contemplar as normas dedicadas a segurança e qualificação dos dispositivos. Com base nisso, este trabalho busca utilizar a inteligência artificial, junto ao aprendizado de máquina, para aplicar testes descritos pelas normas IEC 61000-4-2 – ESD, IEC 61000-4-4 – Burst, IEC 61000-4-5 – Surge e IEC 61000-4-29 - DIPS DC, buscando treinar e otimizar uma rede neural para identificar os tipos de ruídos conduzidos nos microcontroladores sob testes. Após a identificação, ações pró-ativas podem ser tomadas para garantir a robustez esperada do CI. Tais ações podem ser, por exemplo, reduzir a frequência de relógio do CI, aumentar a tensão da fonte de alimentação, utilizar técnicas de redundância no tempo e/ou ativar funções de detecção e correção de erros (EDAC) durante o período em que o sistema estiver operando sob tal exposição ao ruído.

1.1 Objetivos

Este trabalho tem como principal objetivo treinar a rede neural artificial para identificar e classificar tipos de ruídos conduzidos presentes nas linhas de alimentação de circuitos integrados através de técnicas de aprendizado de máquina. Com base na rede neural treinada e classificando os tipos de ruídos, possibilitar um cenário futuro de testes inteligentes integrando execuções autônomas de testes para diversos cenários.

1.1.1 Objetivos Específicos

Especificamente, este trabalho tem como objetivo treinar a rede neural para que ela seja capaz de identificar e classificar os ruídos descritos pelas normas IEC 61000-4-2, IEC 61000-4-4, IEC 61000-4-5 e IEC 61000-4-29, onde cada norma apresenta as seguintes características de testes:

- IEC 61000-4-2 engloba testes de imunidade a descarga eletrostática, com o propósito de simular a descarga eletrostática gerada pela intervenção do corpo humano e avaliar a imunidade do sistema eletrônico, o objetivo é identificar a forma de onda descrita pela norma através do aprendizado de máquina.

- IEC 61000-4-4 especifica testes com rajadas de transientes rápidos com circuito aberto e com carga, acoplados a fonte de alimentação, portas de sinal e aterramento do microcontrolador. O objetivo destes testes é identificar os padrões apontados e descritos na norma através do aprendizado de máquina;
- IEC 61000-4-5 especifica testes para validar a imunidade dos dispositivos quando submetidos a sobretensões. O objetivo é identificar os testes descritos na norma, como tensão de circuito aberto e corrente de curto circuito;
- IEC 61000-4-29 especifica testes com quedas e variações na tensão de alimentação do microcontrolador. O objetivo é identificar através de aprendizado de máquina os testes descritos como *voltage dips*, *short interruptions* e *voltage Variations*;

1.1.2 Escopo do trabalho

É importante definir o escopo deste trabalho, que visa além do citado nos objetivos, criar um *Dataset* (Conjunto de dados) com os valores de tensão e corrente definidos pelas formas de onda das normas citadas.

Os passos mais importantes na criação e desenvolvimento de redes neurais artificiais são a coleta de dados relativos ao problema a ser estudado e a sua separação em um conjunto de dados. Esta tarefa requer uma análise cuidadosa sobre o problema para minimizar ambiguidades e erros nos dados. Além disso, os dados coletados devem ser significativos e cobrir amplamente o domínio do problema, não devem cobrir apenas as operações normais ou rotineiras, mas também as exceções e as condições nos limites do domínio do problema (DE MEDEIROS, 2021).

Portanto, faz parte do escopo deste trabalho, a criação de um conjunto de dados cobrindo os pontos citados acima, a disponibilização deste conjunto em um repositório, bem como a criação da rede neural, treinamento e classificação utilizando a base criada e validando a hipótese deste contexto se tornar uma ferramenta para novos tipos de testes EMC e EMI. O repositório no qual o conjunto de dados e resultados está disponível é o <https://github.com/dougborbabass/dataset-iec-emc-emi>.

1.2 Apresentação dos Capítulos

Essa dissertação foi dividida em três partes e estruturada da seguinte forma:

Parte I – Fundamentos (Capítulos 2 ao 4):

- Capítulo 2: Descreve os conceitos básicos relacionados a área de redes neurais artificiais;
- Capítulo 3: Descreve os conceitos básicos relacionados a área de Tolerância à Falhas.
- Capítulo 4: Descreve os principais conceitos de compatibilidade eletromagnética (EMC) e interferência eletromagnética (EMI), em conjunto a apresentação das principais normas utilizadas neste estudo.

Parte II – Metodologia (Capítulos 5 e 6):

- Capítulo 5: Apresenta detalhadamente a criação das formas de ondas descritas pelas normas e a preparação da base de dados para treinamento da rede neural;
- Capítulo 6: Apresenta a importação da base de dados, interpretação dos dados e criação da rede neural.

Parte III – Resultados e Conclusões (Capítulos 7 e 8):

- Capítulo 7: Apresenta os resultados obtidos durante o desenvolvimento deste trabalho.
- Capítulo 8: Apresenta as conclusões ao desenvolvimento deste trabalho e sugere trabalhos futuros.

2 REDES NEURAIS ARTIFICIAIS

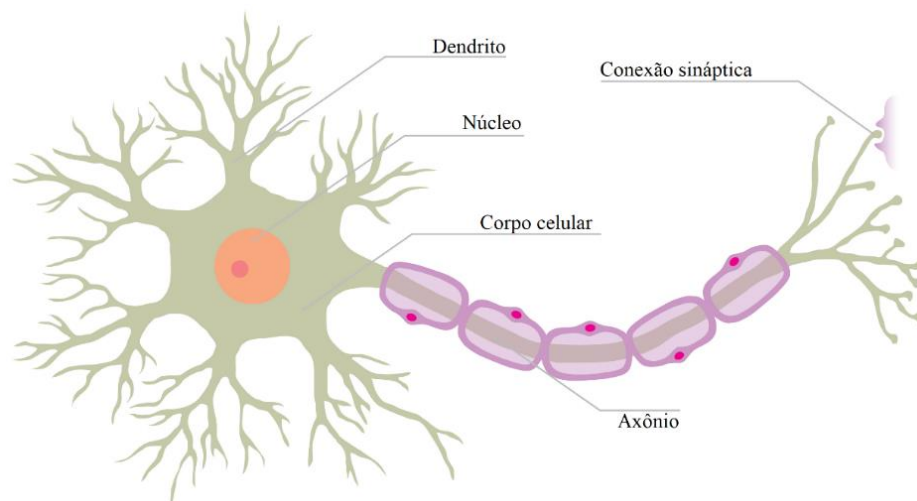
Este capítulo é dedicado a apresentar os principais tópicos relacionados às Redes Neurais Artificiais (RNA's).

2.1 Rede Neural Biológica

Acredita-se que o cérebro humano seja o objeto mais complexo conhecido no universo, sendo composto por cerca de 100 bilhões de neurônios, que se interconectam, formando uma grande malha de neurônios (AMRUTHA; REMYA AJAI, 2018).

O neurônio é responsável pelo gerenciamento de todas as atividades do corpo humano, localizado no cérebro e dividido em três partes básicas: dendritos, corpo celular e axônio (HAYKIN, 2001). Os dendritos são os dispositivos de entrada, tendo como objetivo conduzir os sinais das extremidades para o corpo celular. No outro lado, encontram-se os axônios, que são os dispositivos de saída. As extremidades do axônio de um neurônio são conectadas com os dendritos de outros neurônios através das conexões sinápticas. O neurônio biológico é uma célula que ao ser estimulada em uma entrada gera uma saída que é enviada aos neurônios vizinhos (LAURENE, 1994). A estrutura de um neurônio real é mostrada na Figura 1.

Figura 1 - Estrutura de um neurônio real.



Fonte: Adaptado de (FOUNTAS, 2015).

O cérebro executa tarefas como reconhecimento de padrões e processamento visual em tempos muito mais rápidos do que qualquer computador. Entretanto, o cérebro humano pode ser influenciado pela emoção e não é muito eficiente quando se depara com algum problema que envolve um grande número de variáveis. Além disso, treinar o cérebro humano de forma a que ele execute suas habilidades requer um grande tempo de treinamento (SANTOS, F. L. Dos, 2013).

2.2 Rede Neural Artificial

As RNA's surgem para tentar, em um certo grau, reproduzir o comportamento das redes neurais biológicas, visando utilizar este conhecimento para solucionar problemas onde o nível de complexidade algébrica torna o mesmo inviável de ser resolvido através de métodos convencionais (LORENSI; SANTOS, 2008).

O neurônio artificial é uma estrutura lógico-matemática que procura simular a forma, comportamento e funções de um neurônio biológico. Assim, os dendritos são substituídos por entradas, cujas ligações com o corpo celular artificial são realizadas através de elementos chamados de peso (simulando as sinapses neuronais). Os estímulos são captados pelas entradas e processados pela função de soma e o bias (GUYON, 1991). A modelagem de uma RNA assemelha-se ao cérebro em dois aspectos:

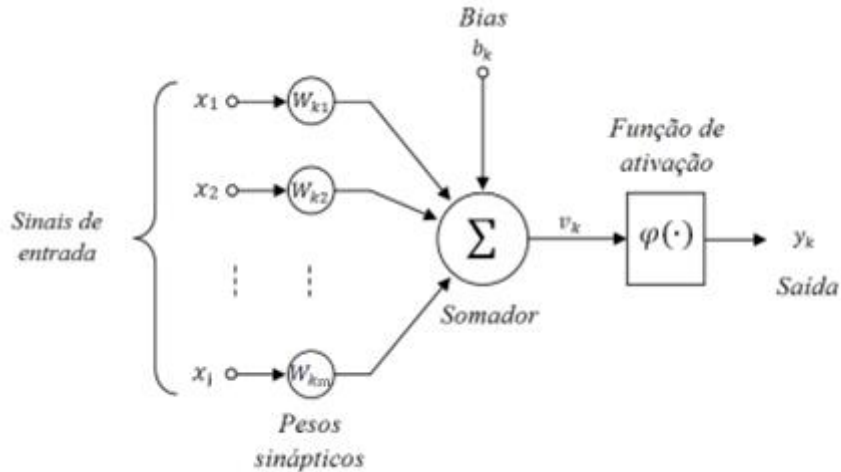
- O conhecimento é adquirido pela rede a partir de seu ambiente através de um processo de aprendizagem;
- Forças de conexão entre neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido.

Outras características importantes das RNA's:

- São modelos adaptativos fáceis de treinar;
- Podem representar domínios complexos (não lineares);
- São capazes de generalização diante de informações incompletas;
- São algoritmos robustos e vastamente utilizados;
- São capazes de fazer armazenamentos associativos de informações.

Diversos pesquisadores propuseram modelos para simular em computadores o funcionamento do cérebro humano, sendo o mais aceito o modelo proposto por McCulloch e Pitts (1943), conhecido como percéptron, mostrado na Figura 2.

Figura 2 - Modelo de neurônio artificial de McCulloch e Pitts (1943).



Fonte: Adaptado de (HAYKIN, 2001).

Analisando matematicamente a Figura 3, tem-se $X = [x_1 \dots x_j]$ como o conjunto de informação que chegam para o percéptron, que são ponderados pelos pesos sinápticos $W_k = [w_{k1} \dots w_{km}]$ com b denominado *bias* sendo todos os sinais unificados pelo função aditiva Σ .

A equação que resulta da combinação linear dos pesos sinápticos w_{ki} com a entrada x_i , denominada atividade pré-sináptica, é dada por:

$$v_k = \sum_{i=1}^j w_{ki}x_i + b_k \quad (2.1)$$

E conseqüentemente, a atividade pós-sináptica, é dada pela função:

$$y_k = \varphi(v_k) \quad (2.2)$$

Onde, φ representa uma função de ativação que processa o somatório das entradas ponderadas pelos seus respectivos pesos e pode assumir diversas formas, com funções lineares e não lineares.

2.3 Funções de ativação

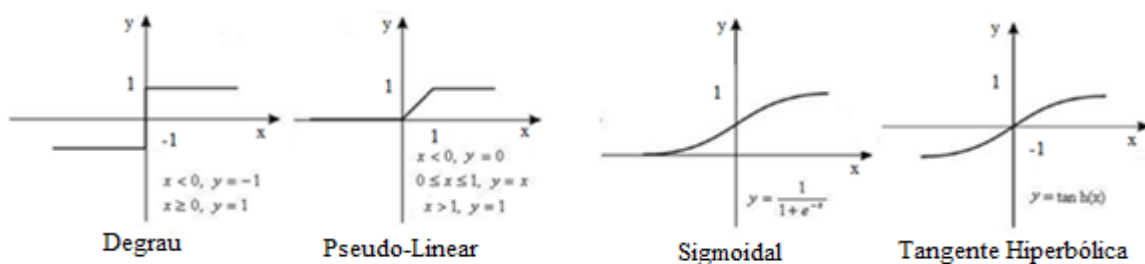
Para a modelagem em redes neurais podem ser utilizadas diferentes funções de ativação. Em teoria, qualquer função contínua e crescente onde o valor de $x \in \mathbb{R}$ e o valor de $y(x, w) \in [-1, 1]$ pode ser utilizada como função de ativação.

Funções de ativação podem ser lineares ou não lineares. Entretanto, em geral as funções trazem componentes de não linearidade em sua saída. Logo, a presença de características dinâmicas nas propriedades envolvidas na definição do estado de ativação tem como consequência a aplicação de equações diferenciais. Caso contrário, não poderiam funcionar durante a retro propagação dos sinais das redes neurais (SANTOS, A. P., 2019).

As funções de ativação são utilizadas para calcular a soma ponderada da entrada e *bias*, que se trata de parâmetros aplicados na decisão de valores para ativação dos neurônios. As funções de ativação também são responsáveis por realizar o processamento dos parâmetros livres, calculando a entrada da camada subsequente e saída da rede neural (NWANKPA *et al.*, 2018).

Os ajustes calculados pelos gradientes são geralmente menores que 1. Conseqüentemente com a multiplicação sucessiva destes resultados, o valor torna-se cada vez menor e tende à zero, reduzindo a ação do gradiente e comprometendo a acurácia do algoritmo de classificação. Por outro lado, se os valores forem maiores que 1, a multiplicação sucessiva aumentará os valores e o gradiente tende ao infinito, ultrapassando os limites do gradiente. Assim, as funções de ativação mantêm os valores desses gradientes em intervalos moderados (NWANKPA *et al.*, 2018). Na Figura 3 são apresentadas algumas das funções mais utilizadas em redes neurais tradicionais.

Figura 3 - Exemplo de funções de ativação tradicionais.



Fonte: Adaptado de (COLAÇO *et al.*, 2007).

Apesar do aceitável desempenho das funções de ativação apresentadas nas arquiteturas de redes neurais tradicionais, as redes neurais profundas em geral apresentam melhores resultados quando utilizadas com as funções de ativação descritas a seguir (SANTOS, A. P., 2019):

2.3.1 Função de ativação logística

A função logística é uma função de ativação clássica utilizada em RNA devido à ela ter representação semelhante da taxa de ativação de um neurônio biológico (HAYKIN, 1998). Existem dois tipos de função logística, que são a função sigmóide e a função tangente hiperbólica (TanH). Por ser uma função clássica ela está presente tanto em arquiteturas de redes neurais profundas e em redes tradicionais. A função sigmóide assume uma faixa contínua de valores de 0 a 1 e é expressa pela seguinte equação:

$$f(u) = \frac{1}{(1 + e^{(-\alpha u)})} \quad (2.3)$$

Onde $u = w_{kj}x_i + b_i$ e α representa o parâmetro de inclinação (LAU; LIM, 2019).

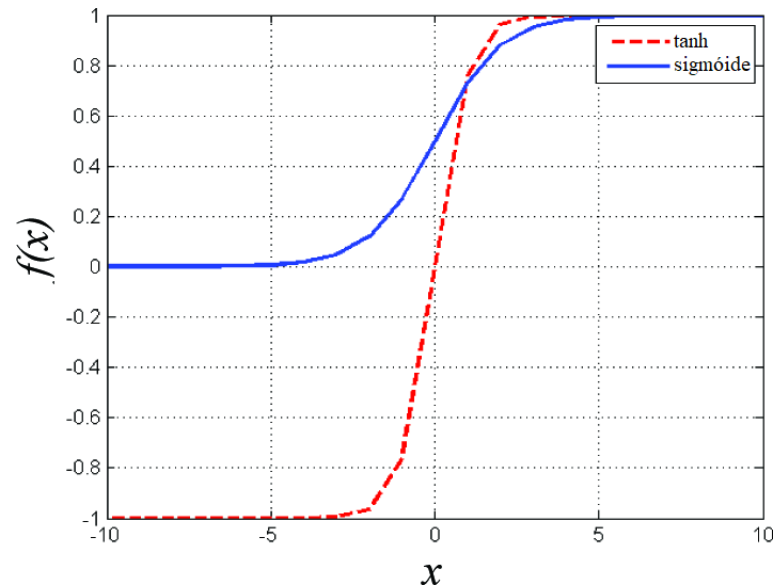
No entanto se olharmos a derivada da função sigmóide, ela satura na parte de seu domínio com valores muito altos ou muito baixos. Com essas derivadas tendendo a zero, a propagação do gradiente desvanece nessas regiões, causando dificuldade no treinamento (GOODFELLOW *et al.*, 2016).

Similar a função sigmoide, a função Tangente Hiperbólica (TanH) também tem um formato de 'S', mas varia de -1 a 1, em vez de 0 a 1 como na sigmoide. A TanH se aproxima mais da identidade, sendo assim uma alternativa mais atraente do que a sigmoide para servir de ativação às camadas ocultas das RNAs e expressa pela equação (GOODFELLOW *et al.*, 2016):

$$f(u) = \frac{2}{(1 + e^{(-2u)}) - 1} \quad (2.4)$$

A Figura 4 mostra a representação gráfica da função sigmóide e também da função tangente hiperbólica.

Figura 4 - Função sigmóide e tangente hiperbólica.



Fonte: Adaptado de (LAU; LIM, 2019).

2.3.2 Função de ativação Softmax

A função de ativação Softmax é utilizada para calcular a probabilidade de distribuição de um vetor de números reais. A função Sigmóide como vimos anteriormente é capaz de lidar apenas com duas classes, por isso para problemas de classificação multi-classe, a função Softmax é uma alternativa. A função Softmax dá a probabilidade de a entrada estar em uma determinada classe e pode ser definida pela equação (NWANKPA *et al.*, 2018):

$$f(u_i) = \frac{e^{(u_i)}}{\sum_j e^{(u_j)}} \quad (2.5)$$

Um exemplo considerando uma classificação de três classes, tendo $u_i = w_{ki}x_i + b_i$ sendo x_i as entradas e w_{ki} seus respectivos pesos e b_i o *bias*, os valores resultantes de u a serem entregues a função de ativação sendo [2.5, 1.5, 0.8], após a função de ativação teríamos as probabilidades de [0.64, 0.24, 0.12], sendo assim a maior probabilidade de ser a classe na posição 0 como a correta na predição.

2.3.3 Função de ativação Linear Retificada (ReLU)

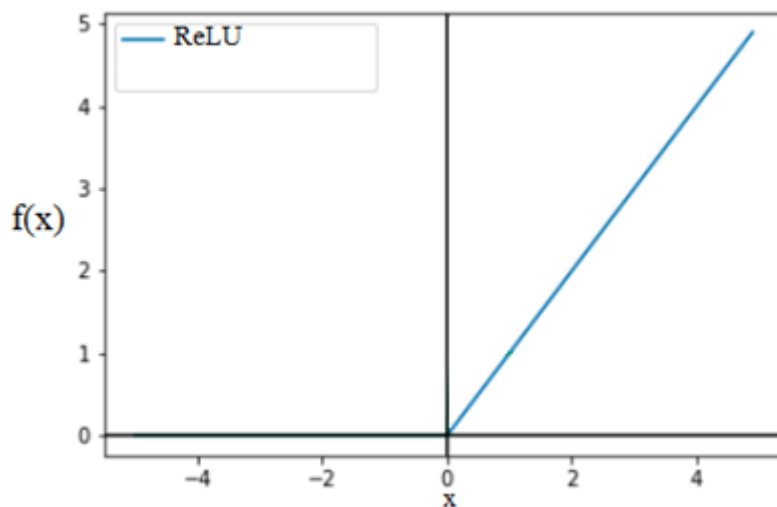
A ReLU é conhecida por ser uma função de ativação de baixo custo computacional, que apesar da simplicidade das operações realizadas obtém bons resultados e, portanto, é amplamente utilizada (NWANKPA et al., 2018).

Redes com a função ReLU são fáceis de otimizar, já que a ReLU é extremamente parecida com a função identidade. A única diferença é que a ReLU produz zero em metade do seu domínio. A ativação ReLU é muito mais eficiente do que as funções sigmoidais vistas anteriormente e é uma das descobertas que contribuiu de forma significativa para a recente popularidade de *Deep Learning*. Essa não linearidade é um ótimo exemplo de como a simplicidade pode ser extremamente poderosa (GOODFELLOW et al., 2016). A função ReLU é expressa pela equação:

$$f(u) = \max(0, u) = \begin{cases} u_i, & \text{se } u_i \geq 0 \\ 0, & \text{se } u_i < 0 \end{cases} \quad (2.6)$$

Esta função retifica os valores das entradas menores que zero, forçando-as a zero e eliminando o problema observado nas funções de ativação Softmax e regressão logística. A função ReLU é comumente utilizada nas camadas ocultas das redes neurais profundas com outra função de ativação aplicada nas camadas de saída da rede (SANTOS, A. P., 2019). A Figura 5 mostra a representação da função ReLU.

Figura 5 - Função ReLU.



Fonte: Adaptado de (RASAMOELINA et al., 2020).

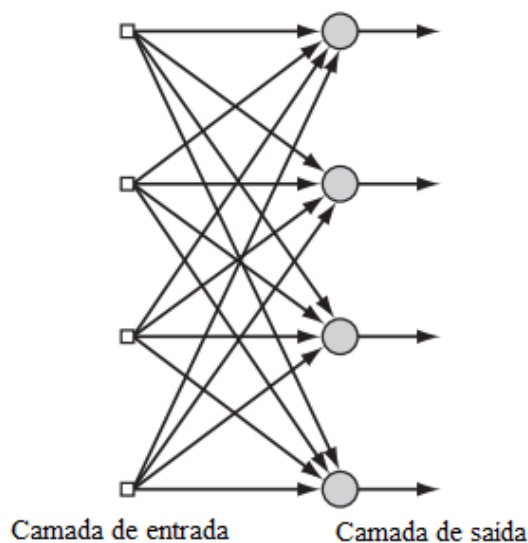
2.4 Arquitetura da rede neural artificial

A questão fundamental na concepção de uma RNA, para uma dada aplicação ou resolução de problema, é a escolha de sua arquitetura. Nessa escolha, define-se a forma como seus neurônios artificiais devem ficar dispostos uns em relação aos outros. Essas disposições são essencialmente estruturadas através do direcionamento das conexões sinápticas dos neurônios artificiais (HAYKIN, 2001).

2.4.1 Rede neural artificial alimentada a diante com camada simples

Também chamada de rede RNA *Feedforward*, tem por característica apresentar uma arquitetura com uma camada de entrada de dados, seguida por uma única camada de neurônios, que é responsável por processar os sinais de entrada e também pela geração do sinal de saída. Nesta arquitetura os sinais são propagados exclusivamente de forma progressiva entre as camadas, ou seja, de modo acíclico. A entrada da rede apesar de ser uma camada, não possui neurônios. Consequentemente, nela não ocorrem os processos computacionais atrelados ao processamento da rede, apenas a captação das características fornecidas pelo vetor de entrada da rede (HAYKIN, 2008). A Figura 6 ilustra uma RNA *feedforward*.

Figura 6 - RNA feedforward com camada única.



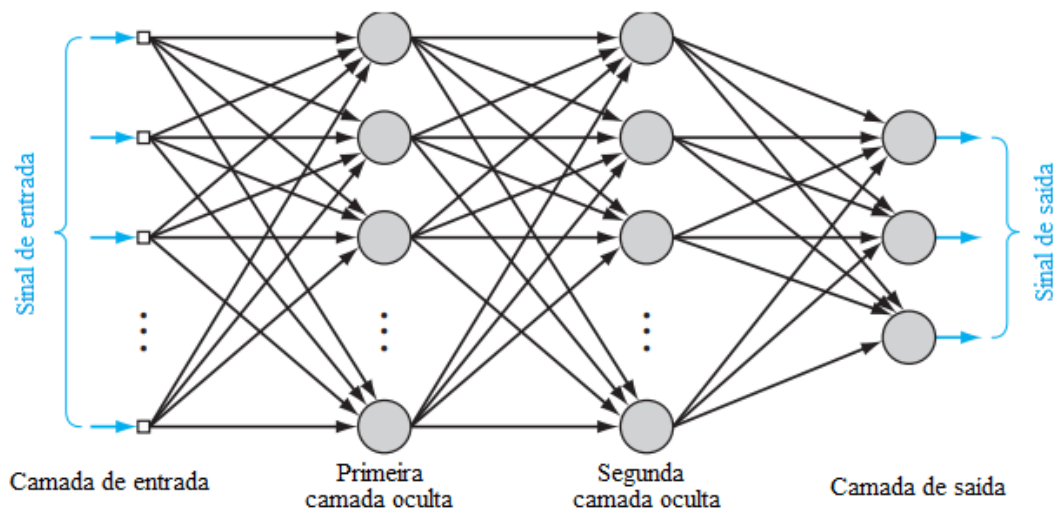
Fonte: Adaptado de (HAYKIN, 2008).

2.4.2 Rede neural artificial com múltiplas camadas

Uma RNA *feedforward* de múltiplas camadas, se distingue pela presença de uma ou mais camadas ocultas e nós computacionais, chamados de neurônios ocultos. Esses neurônios têm a função de intervir entre a entrada externa e a saída da rede de modo útil. Acrescentando uma ou mais camadas ocultas, a rede torna-se apta a extrair estatísticas de ordem elevada de algum desconhecido processo aleatório subjacente, responsável pelo comportamento dos dados de entrada, processo sobre o qual a rede está tentando adquirir conhecimento. A RNA adquire uma perspectiva global do processo aleatório, apesar de sua conectividade local, em virtude do conjunto adicional de pesos sinápticos e da dimensão adicional de interações neurais proporcionada pelas camadas escondidas (HAYKIN, 2008).

Alguns exemplos de redes neurais que utilizam essa arquitetura são: *Perceptron* múltiplas camadas (MLP do inglês *Multilayer Perceptron*) e as redes de funções de base Radial (RBF do inglês *Radial Basis Function*). As redes MLP demonstram grande potencial de aplicação em diversos problemas, tais como, aproximação de funções, classificação, categorização e predição. Essa flexibilidade faz com que este tipo de rede seja aplicado em diversas áreas (SANTOS, A. P., 2019). A Figura 7 apresenta uma MLP *feedforward* com duas camadas ocultas.

Figura 7 - MLP com duas camadas ocultas.



Fonte: Adaptado de (HAYKIN, 2008).

2.5 Processos de aprendizagem

Uma das propriedades mais importante das redes neurais é a habilidade de aprender por intermédio de exemplos e fazer inferências sobre o que aprendeu e, com isso, melhorar gradativamente o seu desempenho. Isso é feito através de um processo iterativo de ajustes aplicado a seus pesos, conhecido como treinamento (HELEGDA, 2014).

O processo de aprendizagem de uma rede se dá pelo ajuste de seus pesos sinápticos, que são os parâmetros livres que se adaptam através de um algoritmo devido a um estímulo aplicado ao sistema (HAYKIN, 1998).

Quanto ao método de aprendizagem, as RNAs se classificam em dois tipos principais: aprendizado supervisionado e aprendizado não-supervisionado.

No aprendizado supervisionado tem-se a existência de um agente externo, que pode ser chamado de professor, responsável por apresentar um conjunto de amostras de entrada e a respectiva saída desejada, analisando a saída calculada pela rede e comparando-a com a saída desejada. Caso a saída calculada pela rede não seja igual ou próxima ao valor desejado, os pesos sinápticos e níveis de bias precisam ser ajustados iterativamente, sob a influência do sinal de erro, de forma a minimizar a diferença entre a saída gerada pela rede e a saída desejada (ALBERTO; SILVA, 2015). A cada interação, em virtude do conhecimento prévio das saídas de cada entrada, o professor é capaz de fornecer à rede neural uma resposta desejada para o respectivo elemento do conjunto de dados. A resposta do professor representa a ação esperada pela rede neural (HAYKIN, 2008). A desvantagem principal deste tipo de aprendizado é que se necessita conhecer os valores de resposta para cada padrão apresentado à RNA (BALESTRASSI, 2000).

Já no aprendizado não supervisionado, não há necessidade de um agente externo para verificar o processo de aprendizado. No aprendizado não supervisionado a RNA é desprovida de uma saída desejada para ser utilizada no melhoramento de seu comportamento e o conjunto de treinamento consiste de somente entradas. A rede tenta, nesse caso, descobrir aspectos comuns entre as entradas separando os padrões entre si (BALESTRASSI, 2000). Durante esse aprendizado, verifica-se a existência de regularidades nos dados de entrada, por meio de suas características estatisticamente mais relevantes (BRAGA; CARVALHO; LUDERMIR, 2007). O término do método de aprendizagem se dá, basicamente, quando a RNA encontra uma solução para todas as entradas fornecidas. Entretanto, outros parâmetros, também, podem ser levados em consideração, como, por exemplo, a definição de uma taxa de erro a ser atingida ou o número máximo de ciclos (ALBERTO; SILVA, 2015).

2.6 Algoritmo de aprendizagem por correção de erro

O conceito de aprendizagem pode ser definido como a modificação adaptativa dos parâmetros de uma rede neural por meio dos estímulos originários do ambiente. O modelo de aprendizagem, por sua vez, é o processo intrínseco por onde os ajustes dos parâmetros acontecem (HAYKIN, 2008).

O Treinamento de uma rede MLP, consiste em ajustar os pesos e os limiares de disparo de suas unidades para que a classificação desejada seja obtida. Quando um padrão é inicialmente apresentado à rede, ela produz uma saída e, após medir a distância entre a resposta atual e a desejada, são realizados os ajustes apropriados nos pesos de modo a reduzir esta distância (HAYKIN, 2001). O algoritmo mais utilizado para o treinamento destas redes MLP é uma generalização da Regra Delta denominado de Algoritmo *Backpropagation* (MIGUEZ, 2012). Esse ajuste nos pesos ocorre por meio de um processo de retro-propagação que pode ser dividido em duas fases (SILVA, 1998):

- ***Feedforward***: *Feedforward* pode ser definido como a fase de propagação do sinal funcional entre as camadas da rede. Nesta etapa, os valores provenientes do ambiente são obtidos como o sinal de entrada e modificados pela rede enquanto transitam pelas camadas até gerar o sinal de saída. Caso a saída da rede seja a esperada pelo professor, os valores dos pesos sinápticos serão mantidos, caso contrário, será ativado o processo de retro-propagação do erro (*Backpropagation*) (SANTOS, A. P., 2019)
- ***Backpropagation***: O algoritmo *Backpropagation* tem o intuito de proporcionar às redes neurais de múltiplas camadas uma forma de extrair conhecimento por meio de seus erros de classificação. Esta fase ocorre quando a saída da rede é diferente da esperada pelo professor. Quando este erro acontece, o sinal de erro é calculado de modo a ajustar a distância entre a resposta da rede e a saída esperada (SANTOS, A. P., 2019). Por fim os pesos são ajustados de forma que a distância entre a resposta da rede e a saída esperada seja reduzida.

Existem várias abordagens para construção de algoritmos de aprendizado. Entretanto, uma das abordagens mais bem-sucedidas é a aprendizagem baseada em gradientes (LECUN *et al.*, 1998).

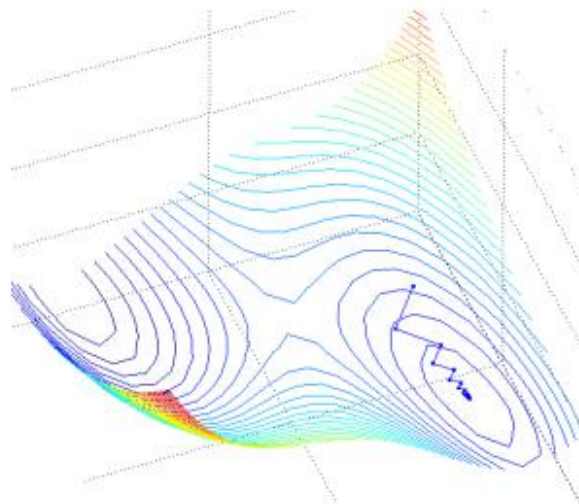
2.6.1 Gradiente Descendente

O gradiente descendente é um algoritmo de otimização (RUMELHART; HINTON; WILLIAMS, 1986), que funciona de forma iterativa, onde tenta aproximar uma função para o mínimo local alterando os seus parâmetros a cada passo. Esta iteração pode ser definida pela equação abaixo (SANTOS, A. P., 2019 apud DENIS; SCHANABEL, 1983):

$$w_{t+1} = w_t - \gamma \frac{1}{n} \sum_{i=1}^n \nabla_w f(z_i, w_t) \quad (2.7)$$

Onde, w_{t+1} representa o valor do peso na próxima interação da rede, ajustado com base no peso atual denotado por w_t . γ representa um valor escalar aplicado ao cálculo do ajuste que deve ser adequadamente escolhido a partir de suposições de regularidade; quando a estimativa inicial de w estiver perto o suficiente do ideal, e quando o ajuste γ é suficientemente pequeno, este algoritmo consegue atingir uma convergência linear (SANTOS, A. P., 2019 apud DENIS; SCHANABEL, 1983). O número total de atributos da entrada aplicados na ponderação é representado por n . z representa um par obtido pela entrada da rede (x) e a saída esperada (y), ou seja, $z = (x, y)$. Portanto a função de perda é determinada por $\varrho(z_i, w_t)$. A Figura 8 mostra o exemplo de uma superfície de erro e os pontos dos gradientes descendentes calculados até atingir o valor de mínimo da superfície.

Figura 8 - Superfície de erro de uma rede MLP e a trajetória do gradiente descendente.



Fonte: (LORENSI; SANTOS, 2008).

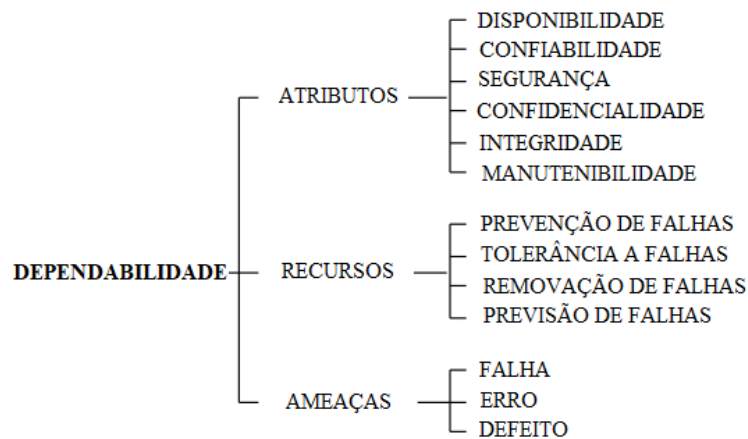
3 CONCEITOS BÁSICOS DE TOLERÂNCIA A FALHAS

3.1 Introdução

A busca por sistemas mais sofisticados e com um poder tecnológico maior tornou mais complexo os sistemas e com isso sujeito a falhas. Com essa sofisticação, distribuição, dinamismo e complexidade dos sistemas, há uma necessidade de torná-los mais seguros e tolerantes a falhas (RODRIGUES, 2014).

Um sistema para ser Tolerante a Falhas (TF) tem que continuar fornecendo seus serviços mesmo diante de falhas. A TF é o conjunto de técnicas utilizadas para detectar e/ou mascarar e tolerar falhas no sistemas (SIQUEIRA; ABDELOUAHAB, 2006). A indicação da qualidade e confiança desses sistemas é denominada **dependabilidade**, uma tradução literal do termo em inglês *dependability* (SOARES, 2021). O conceito de dependabilidade é amplo e diferentes propriedades podem ser colocadas em ênfase (LAPRIE, J., 1995), a Figura 9 ilustra o desdobramento dessas propriedades.

Figura 9 - Diagrama de dependabilidade.



Fonte: Adaptado de (LAPRIE, J., 1995)

Algumas definições didáticas de sistemas e suas características (AVIŽIENIS *et al.*, 2004):

- **Sistema:** é uma entidade que interage com outras entidades, ou seja, outros sistemas, incluindo hardware, software, humanos e o mundo físico com seus fenômenos naturais;
- **Função de Sistema:** é o que o sistema se destina a fazer e é descrito pela especificação funcional em termos de funcionalidade e desempenho;

- **Comportamento:** é o que o sistema faz para implementar sua função e é descrito por uma sequência de dados;
- **Serviço:** o serviço entregue por um sistema (em função de seu provedor) é seu comportamento e é identificado por outro sistema de iteração, o usuário.

3.2 Atributos

A definição original da dependabilidade é a capacidade de entregar serviços que possam ser confiáveis, e o critério para decidir se o serviço é confiável é a capacidade do sistemas evitar defeitos de serviço mais frequentes e mais severos que o aceitável (AVIŽIENIS *et al.*, 2004). Os mais recorrentes conceitos de dependabilidade são apresentados na sequência (LAPRIE, J., 1995):

- **Disponibilidade:** é a prontidão do sistema estar operacional quando necessário;
 - **Confiabilidade:** é a habilidade do sistema em apresentar continuidade na realização de um serviço corretamente
 - **Segurança:** é a não ocorrência de consequências catastróficas em relação ao ambiente e o usuário;
 - **Confidencialidade:** é a não ocorrência de divulgação de informações não autorizadas;
- Manutenibilidade:** é a capacidade do sistema de receber reparos e evoluções.

3.3 Recursos

O desenvolvimento de um sistema confiável requer a utilização de um conjunto de técnicas e métodos que podem ser classificados como (LAPRIE, J., 1995):

- **Prevenção de Falhas:** métodos para prevenir a ocorrência de falhas;
- **Tolerância a Falhas:** métodos para garantir um serviço do sistema mesmo com a ocorrência de uma falha;
- **Remoção de Falhas:** métodos para reduzir a presença (número, gravidade) das falhas;

- **Previsão de Falhas:** métodos para estimar o presente número, a incidência e as futuras consequências de falhas.

A prevenção e a tolerância a falhas, dizem respeito a capacidade do sistema em entregar um serviço que seja confiável. Por outro lado, a remoção e a previsão de falhas visam alcançar a confiança nessa capacidade, garantindo que o sistema seja funcional dentro de suas especificações (AVIŽIENIS *et al.*, 2004).

3.3 Ameaças

Na literatura se encontra inúmeras descrições de como projetar um sistema tolerante a falhas, e para se projetar um sistema é preciso primeiramente entender com as falhas ocorrem. Em um sistema os níveis de percepção podem ser: **falha, erro e defeito** (RODRIGUES, 2014).

A seguir, são apresentados detalhes sobre os conceitos relacionados as ameaças:

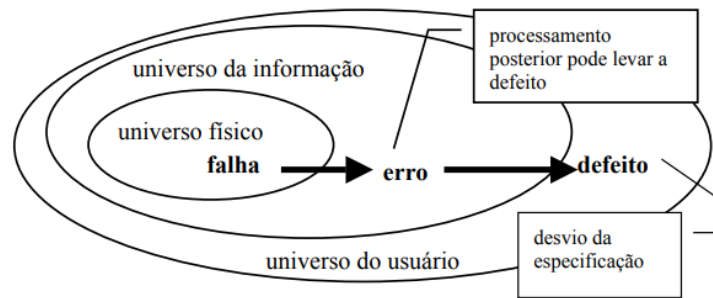
Falha: é definida como a causa física ou algorítmica do erro (WEBER, 2003). As constantes interferências externas, problemas de implementação, componentes defeituosos assim como o intrínseco envelhecimento dos componentes, são as principais causas de falhas (BENFICA, 2007). Além da causa, para definir uma falha considera-se ainda (WEBER, 2003):

- a) **Natureza:** falha de *hardware*, falha de *software*, de projeto, de operação, entre outros;
- b) **Duração:** permanente ou temporária (intermitente ou transitória);
- c) **Extensão:** estabelece a propagação da falha, determinando se é local a um módulo, ou global
- d) **Valor:** se a falha é determinada ou indeterminada no tempo.

Erro: Sendo o serviço uma sequência de estados do sistema, um defeito no serviço significa que pelo menos um estado externo do sistema desviou do estado correto. Um erro que afeta o serviço é uma indicação de que falha ocorre ou ocorreu (LAPRIE, J., 1995). Quando ocorre um erro, o período até a manifestação do defeito devido ao erro é chamado de **latência de erro** (WEBER, 2003).

Defeito: é definido como um desvio da sua especificação. Defeitos não podem ser tolerados, mas deve ser evitado que o sistema apresente defeito (WEBER, 2003). Ilustrando os conceitos apresentados acima, a Figura 10 apresenta uma simplificação desses conceitos.

Figura 10 - Modelo dos 3 universos: Falha, Erro e Defeito.



Fonte: (WEBER, 2003).

Exemplificando, podemos dizer, por exemplo, que a constante oscilação da energia elétrica em nossas casas pode ser considerada uma **falha**, e essa falha, por sua vez, pode gerar um **erro** em nosso computador, fazendo com que ocorra uma troca inesperada de bits. Por consequência, seria gerado um **defeito**, podendo travar o equipamento ou até mesmo provocar a queima do computador, afetando assim sua disponibilidade (DANTAS, 2013).

As falhas de software e também de projeto são consideradas atualmente um dos maiores problemas em computação crítica. Sistemas críticos, tradicionalmente, são construídos de forma a suportar falhas físicas. Assim é compreensível que falhas não tratadas, e não previstas, sejam as que mais incomodem, pois possuem um grande potencial de comprometer a confiabilidade e disponibilidade do sistema (WEBER, 2003). A Tabela 1 apresenta um levantamento de fontes de defeitos e a frequência com que ocorrem.

Tabela 1 - Causas usuais de defeitos em sistemas de computação.

Não Tolerantes a Falhas	Tolerantes a Falhas
MTTF: 6 a 12 semanas	MTTF: 21 anos
Indisponibilidade após defeito: 1 a 4 horas	
Fontes de Defeitos:	Fontes de Defeitos:
hardware 50%	software 65%
software 25%	operações 10%
ambiente 15%	hardware 8%
operações 10%	ambiente 7%

Fonte: Adaptado de (LAPRIE, J. C., 1998).

4 COMPATIBILIDADE ELETROMAGNÉTICA

4.1 Introdução

A utilização de dispositivos eletrônicos em diversas áreas teve um aumento exponencial no decorrer dos últimos anos, tornando-os indispensáveis em nosso dia a dia (SOARES, 2021). Um exemplo que merece atenção, é o grande aumento no desenvolvimento de satélites de pequeno porte (micro- e nano-satélites), que devido ao baixo custo proporcionado pelo uso de componentes COTS (*component-of-the-shelf*) tem sido produzidos em larga escala por universidades, centros de pesquisa e pequenas empresas ao redor do mundo. Neste caso específico de aplicação, a eletrônica embarcada deve apresentar resiliência não somente à interferência eletromagnética, mas também aos efeitos combinados de desta com a radiação ionizante (total-ionizing dose: TID e single-event upsets: SEUs). (BENFICA, 2020), (GOERL, 2019).

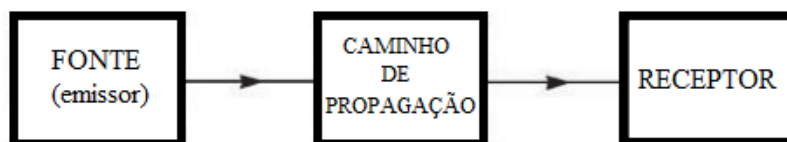
Fenômenos envolvendo emissão de ondas eletromagnéticas são conhecidos desde o princípio da utilização de equipamentos de comunicação que utilizam eletricidade. Os fenômenos que envolvem emissões eletromagnéticas podem ser gerados de forma proposital (como por exemplo, transmissores de radiofrequência) ou de forma natural (como por exemplo, raios). Outros equipamentos, como por exemplo computadores, que têm sua operação baseada em pulsos digitais, também podem gerar emissões eletromagnéticas (PRYTULA, 2020).

Essas emissões eletromagnéticas são capazes de causar mau funcionamento em equipamentos instalados próximos as fontes geradoras de emissão devido à interferência gerada, por conta disto, se não considerados os devidos cuidados em relação à compatibilidade eletromagnética (EMC, do inglês, *Electromagnetic Compatibility*), podem ocorrer problemas nos equipamentos levando a falhas e degradação da confiabilidade. Um equipamento é compatível eletromagneticamente se satisfaz três critérios de operação, conforme (PAUL, 2006):

- a) Não causa interferência eletromagnética (EMI, do inglês, *Electromagnetic Interference*) em outros sistemas;
- b) Não é suscetível a emissões (EMI) geradas por outros sistemas;
- c) Não causa interferência (EMI) em si próprio.

A EMC está relacionada com a geração, transmissão e recepção da energia eletromagnética. Assim, o problema básico da EMC está mostrado na Figura 11.

Figura 11 - Problema básico de EMC



Fonte: Adaptado de (PAUL, 2006).

Deste modo, uma fonte produz a emissão e um caminho de propagação transfere a energia do emissor para um receptor, onde a energia é processada resultando em um comportamento desejado ou indesejado. A interferência ocorre se a energia recebida faz com que o receptor se comporte de modo indesejado. A transmissão ou recepção não intencional da energia eletromagnética não é necessariamente má, o comportamento indesejado do receptor é que constitui a interferência. O processamento da energia recebida pelo receptor é que define a existência ou não da interferência (LIZ, 1999). Então, para prevenir problemas de interferência, trata-se a questão de três maneiras (PAUL, 2006):

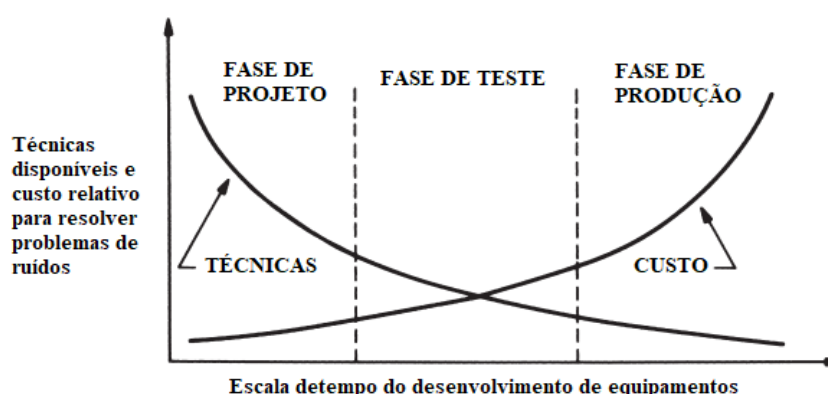
- a) Reduzir as emissões indesejadas na fonte;
- b) Tornar o caminho de propagação o mais ineficiente possível;
- c) Tornar o receptor o menos possível suscetível às emissões

Para conseguir estes objetivos, uma primeira alternativa, é suprimir tanto quanto possível a emissão na fonte. Por exemplo, os tempos curtos de subida e descida de sinais de comando, causam elevado conteúdo espectral e geram mais componentes de alta frequência e componentes de alta frequência aumentam a eficiência do caminho de propagação ou acoplamento. Desta forma, aumentar os tempos de subida e descida é uma alternativa (PRYTULA, 2020).

Quando abordamos as principais vítimas da EMI nos sistemas, na maioria das vezes, os circuitos integrados (CIs) estão presentes, sendo mais susceptíveis a danos causados por condições de sobretensões ou sobrecorrentes. Mesmo que não sejam danificados, um acoplamento de interferência em seus pinos de alimentação, já seria suficiente para causar um mau funcionamento (SOARES, 2021 apud BEN DHIA *et al.*, 2006).

O projeto de equipamentos baseados em todos os critérios de EMC é o ponto mais importante para evitar retrabalho e o alto custo para correção. Quando o projetista segue todos os critérios de projeto baseados em EMC, a probabilidade de acerto do projeto fica em torno de 90% (OTT, 2009). Na Figura 12, pode-se ver um gráfico relacionando o emprego de técnicas para resolver possíveis problemas durante a fase de projeto em relação a fase já em produção do produto.

Figura 12 - Relação custo *versus* técnicas utilizadas em um projeto com EMC.

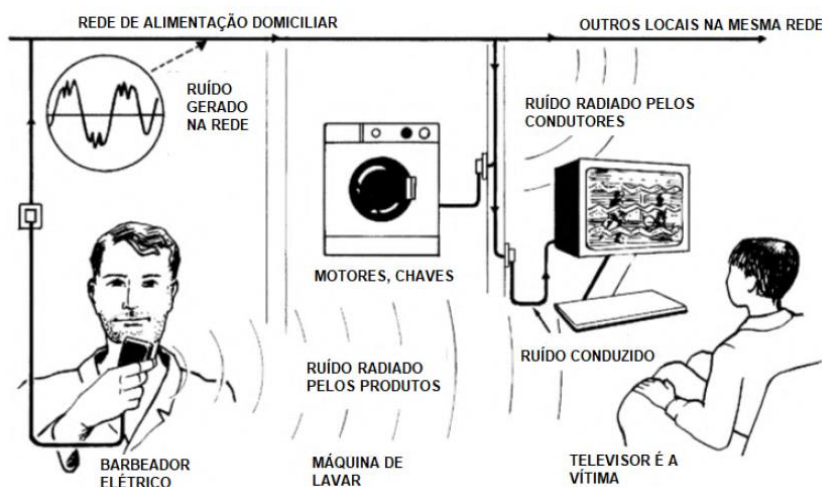


Fonte: Adaptado de (OTT, 2009).

4.2 Ambiente Eletromagnético

Um ambiente eletromagnético (EME, do inglês, Electromagnetic Environment) pode ser definido por vários elementos, tais como a rede de energia elétrica, o tipo de edificação ou local de instalação do equipamento, outros equipamentos eletrônicos instalados e até mesmo o ambiente externo (BENFICA, 2007). A análise de qualquer alteração do ambiente eletromagnético, vinda de operações ou conexões de sistemas eletrônicos é extremamente importante para a garantia do conceito de EMC (MORAES, 2008), visto que, quaisquer alterações provocariam impactos na distribuição dos campos eletromagnéticos nele inseridos e, conseqüentemente alteraria as características EMC dos equipamentos (SOARES, 2021). A Figura 13 ilustra um cenário típico de um ambiente eletromagnético.

Figura 13 - Ambiente eletromagnético.



Fonte: Adaptado de (MORGAN, 1994).

4.3 Interferência Eletromagnética (EMI)

As interferências eletromagnéticas são perturbações causadas a um circuito, dispositivo ou sistema eletrônico por radiações eletromagnéticas emitidas a partir de uma fonte externa (IEC, 1990). Os efeitos dessas interferências podem provocar no circuito “vítima” a limitação de seu desempenho, como também a interrupção temporária das suas funções. É interessante notar que um circuito considerado “vítima”, também pode ser considerado um “agressor”, pois ele também é fonte geradora de ruído irradiado para os seus circuitos vizinhos na placa (ESTEVAN, 2021).

Como visto anteriormente na Figura 14, existem diversas fontes de geração de ruído presentes no nosso dia a dia, essas fontes ainda podem ser divididas em naturais e não naturais. As fontes naturais podem ser provenientes de descargas elétricas, ruídos cósmicos provocados por explosões solares. No caso de quedas de raios sobre a rede de distribuição de energia elétrica, o distúrbio é propagado pelos fios até a instalação interna, podendo provocar diversos danos. As fontes de EMI não naturais são geradas tanto dentro do ambiente em que o equipamento está instalado, como fora dele, exemplos dessas fontes não naturais podem ser os acionamentos de cargas indutivas em motores elétricos, cargas resistivas de aquecedores, lâmpadas fluorescentes, equipamentos que utilizam radiofrequência (RF), aparelhos de micro-ondas, equipamentos de comunicação móvel, etc (BENFICA, 2007).

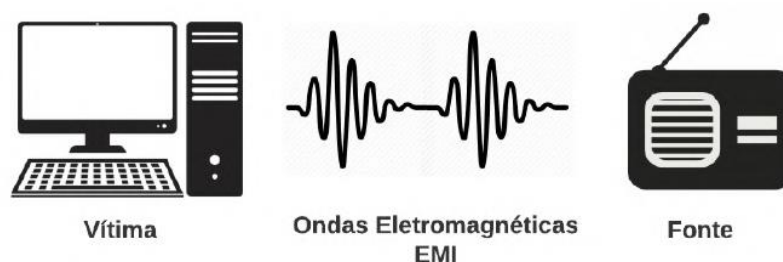
Independente se natural ou não, a qualidade da energia elétrica é comprometida e conseqüentemente, acaba gerando diferentes fenômenos que podem ser prejudiciais ao usuário se não tomadas as medidas corretas (SOARES, 2021), (SOARES, 2021a).

4.4 EMI Radiada e EMI Conduzida

Como visto anteriormente, as emissões eletromagnéticas são fenômenos nos quais a energia de uma determinada fonte geradora flui para outro dispositivo ou circuito “vítima”. Quando analisamos o meio de propagação, esses fenômenos são divididos em dois tipos: **interferência radiada** e **interferência conduzida**.

Na **Interferência radiada**, tem-se um elemento irradiando um campo eletromagnético (fonte de interferência), que se propaga na atmosfera e atinge equipamentos eletrônicos e sistemas de telecomunicação (receptores de interferência) podendo-os levar a falhas ou ainda interrupções de suas funções (KRAUS; FLEISCH, 1999). A Figura 14 apresenta um exemplo de EMI radiada.

Figura 14 - EMI radiada.

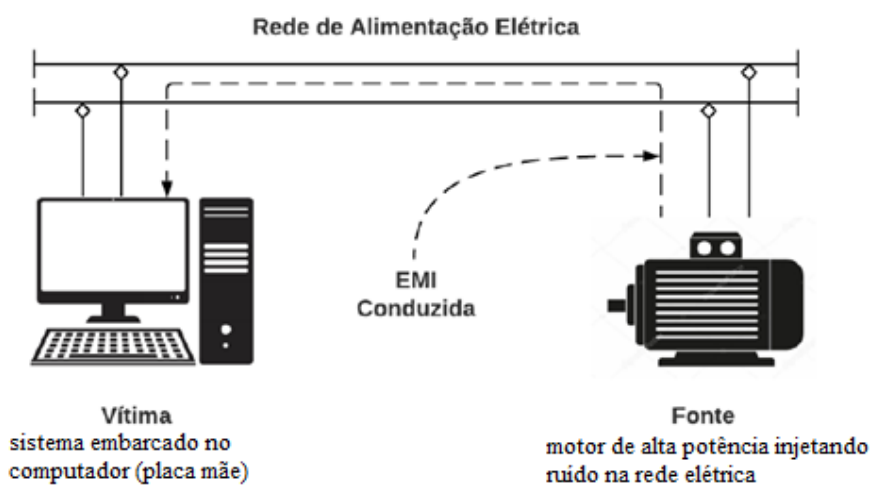


Fonte: (SOARES, 2021)

Normalmente essa irradiação ocorre quando um ou mais elementos da fonte geradora, como cabos e trilhas de circuito impresso comportam-se como antenas (PRESTES, 2010).

A **Interferência conduzida** normalmente ocorre com mais frequência nas redes elétricas de sinal alternado (KRAUS; FLEISCH, 1999). Conseqüentemente como essa rede alimenta vários sistemas ruidosos (motores, comutadores, etc.), estes ruídos são conduzidos pela rede elétrica podendo interferir em outros sistemas, equipamentos e circuitos que são alimentados por essa mesma rede (BENFICA, 2007). Nessa perturbação, a energia é propagada da fonte de interferência ao sistema “vítima” através de condutores de alimentação elétrica, terminais de dados I/O (do inglês, *Input/Output*) ou até mesmo pela referência de tensão (PRESTES, 2010). A Figura 15 mostra um exemplo de EMI conduzida ocorrendo pela rede elétrica, mostrando uma máquina elétrica afeta o comportamento de equipamento eletrônico. Este tipo de acoplamento é muito comum em ambientes industriais, comerciais e domésticos (BENFICA, 2007).

Figura 15 - EMI conduzida.



Fonte: Adaptado de (SOARES, 2021).

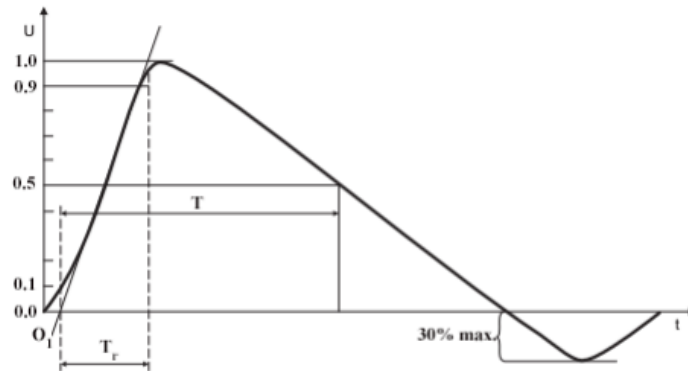
4.4.1 Tipos de EMI Conduzida

Ambientes compartilhados por diferentes equipamentos eletrônicos acabam propiciando um cenário favorável a propagações de radiações eletromagnéticas indesejadas. Quando se fala de imunidade conduzida, as normas estabelecem que os sistemas devem ser imunes a cinco fenômenos fundamentais (CAROBBI *et al.*, 2013):

1. Surtos (*Surge*);
2. Descargas eletrostáticas (ESD, do inglês *Electrostatic Discharge*);
3. Transientes elétricos rápidos de tensão (EFT, do inglês *Electrical Fast Transients*);
4. Quedas e variações de tensão;
5. Ruído conduzido.

Os **surtos** são definidos como um sinal de tensão, corrente ou potência, que se propaga ao longo da rede elétrica ou circuito, caracterizado por um rápido aumento seguido de uma declinação lenta (IEC, 1990). A Figura 16 exemplifica uma forma de onda de tensão que representa um fenômeno eletromagnético do tipo surto.

Figura 16 - Exemplo de forma de onda de uma variação de tensão do tipo surto.

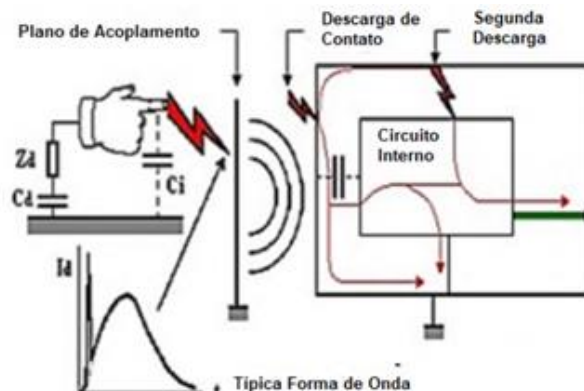


Fonte: (IEC, 2005)

A origem dos surtos ainda pode ser distinguida por chaveamento de cargas de potência e também por fenômenos naturais, como por exemplo, relâmpagos (IEC, 2005). Os surtos por chaveamento de potência são causados pela indutância do chaveamento instantâneo de cargas de corrente na alimentação (OTT, 2009), Já os relâmpagos, produzem surtos de tensão por descargas atmosféricas diretas ou indiretas, ou por um fluxo do distúrbio do aterramento (SOARES, 2021).

A **descarga eletrostática** é uma transferência de carga elétrica entre corpos de diferentes potenciais eletrostáticos próximos ou através de contato direto (IEC, 2001). Quando corpos eletricamente carregados entram em contato ou ficam próximos de corpos com potencial elétrico diferentes, as cargas migram de um corpo ao outro, a fim de nivelar o potencial. A amplitude e a forma gerada depende da diferença de potencial e da impedância total envolvida (BUZDUGAN; BALAN, 2019). A Figura 17 exemplifica o acoplamento eletromagnético de dois sistemas que gera uma descarga eletrostática.

Figura 17 - Mecanismo ESD.



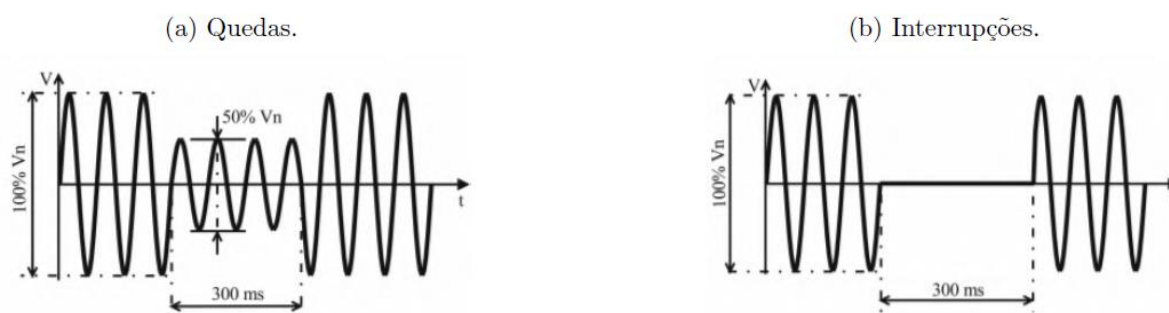
Fonte: Adaptado de (BUZDUGAN; BALAN, 2019).

Os **transientes elétricos rápidos** (EFT), representam uma sequência de pulsos ou oscilação de duração limitada. São transitórios rápidos de tensão que muitas vezes são gerados por perturbações em redes de energia de baixa tensão, arcos elétricos resultantes da comutação de contatos de relés e diferentes interruptores, desligamento de cargas indutivas e lâmpadas fluorescentes (BUZDUGAN; BALAN, 2019). Estes distúrbios podem ser de curta duração na ordem de nano ou microssegundo, ou até mesmo mais longos, na ordem de milissegundos. Os níveis de energia de tais eventos transitórios podem variar em uma ampla gama também, incluindo níveis de energia relativamente baixos que podem interromper a operação normal de circuitos eletrônicos sensíveis, ou perturbações com amplitudes altas o suficiente para completamente destruir todo o sistema eletrônico (BAUER; DEUTSCHMANN; WINKLER, 2015).

Apesar da similaridade com o fenômeno de surtos, os EFTs são distúrbios de alta frequência e baixa energia resultante de cargas indutivas, enquanto os surtos, são distúrbios de alta frequência e alta energia, provenientes de cargas capacitivas (CAROBBI *et al.*, 2013).

A **queda de tensão** tem por característica uma redução súbita na tensão de alimentação dos sistemas, seguida do seu retorno ao valor nominal. As **interrupções** são os desligamentos momentâneos por curto períodos normalmente não superior a 1 minuto. Na prática, uma descida na amplitude de pelo menos 80% da tensão nominal pode ser considerada uma interrupção. **Variação de tensão** é a mudança gradual da tensão de alimentação para um valor superior ou inferior à tensão nominal, com duração longa ou curta (IEC, 2000). Esses distúrbios são causados principalmente pelo chaveamento de cargas de alta potência, partida de motores ou sobrecarga de sistemas. A Figura 18 ilustra uma representação de uma queda de tensão e uma interrupção.

Figura 18 - Ilustração de queda e interrupção na tensão.



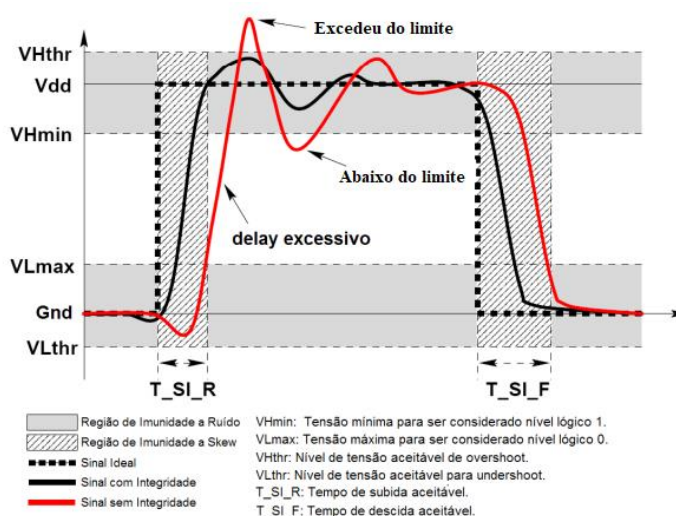
Fonte: Adaptado de (PÉREZ; DONSIÓN, 2003).

A **imunidade para ruído conduzido** está relacionada aos campos eletromagnéticos radiados, que produzem tensões induzidas nos cabos de alimentação do sistema. Essas perturbações podem ser provenientes de fontes intencionais, como as emissões de antenas, TVs e celulares, e também de fontes não intencionais, como por exemplo motores e conversores (RAIZER, 2005).

4.5 Efeitos da Interferência Eletromagnética na Eletrônica

A integridade de um sinal é definida como a característica e/ou habilidade de um determinado sinal gerar respostas corretas em um circuito e/ou sistema eletrônico (NOURANI; ATTARHA, 2002). Um sinal com uma boa integridade apresenta, entre as suas características, níveis de tensão e tempos de transição de acordo com as exigências e especificações de projeto do sistema em que este sinal está associado. A Figura 19 apresenta um exemplo de sinal e suas características de integridade.

Figura 19 - Exemplo de integridade de sinais.



Fonte: Adaptado de (NOURANI; ATTARHA, 2002).

Como abordado nos capítulos anteriores, a coabitação de equipamentos de diferentes tecnologias no mesmo ambiente, contribuem para a emissão e/ou susceptibilidade de equipamentos com tecnologia SoC de alto desempenho e a interferências eletromagnéticas, potencializando assim, a possibilidade de ocorrer distúrbios como (MORAES, 2008):

- a) **Flutuações nas linhas de alimentação:** reduções momentâneas nos níveis de tensão das linhas de alimentação de um dispositivo e/ou sistema eletrônico;
- b) **Ground Bounce:** elevações momentâneas dos níveis de tensão das linhas de referência de tensão de um dispositivo e/ou sistema eletrônico;
- c) **Ruído das Linhas de Alimentação:** variações de corrente de carga provenientes do rápido chaveamento de circuitos;
- d) **Skew:** diferenças nos tempos de propagação de dois ou mais sinais transmitidos simultaneamente através da rede de distribuição de um circuito e/ou sistema.

Neste cenário, a degradação dos sinais de dispositivos e/ou sistemas eletrônicos é um agente decisivo na ocorrência de falhas funcionais e distúrbios nos mais diversos componentes dos sistemas. (MORAES, 2008).

4.6 Normas e Regulamentações

Durante a década de 80, os fenômenos relacionados a EMC, ligados a emissões conduzidas ou radiadas dos sistemas eram a principal preocupação. Com o passar do tempo, o aumento significativo de diversas tecnologias, tornou mais frequente os relatos de fenômenos relacionados a EMI, tornando necessário alocar e proteger o espectro eletromagnético, principalmente do setor de comunicações (SOARES, 2021). Dessa forma, na década de 90, o conceito de imunidade eletromagnética ganhou destaque, forçando os governos e comitês internacionais a estabelecerem testes e requisitos normativos para avaliação desses sistemas (MORGAN, 1994).

A *International Electrotechnical Commission* (IEC) foi criada em junho de 1906 na Inglaterra e é, nos dias de hoje, uma das principais organizações mundial responsável no desenvolvimento e publicações de normas técnicas internacionais para as áreas relacionadas a engenharia elétrica e eletrônica (IEC, [s. d.]).

A IEC reúne todas as tecnologias relacionadas à eletrônica, incluindo magnetismo e eletromagnetismo, eletroacústica, telecomunicações, geração e distribuição de energia elétrica. Também trata sobre terminologia e símbolos, compatibilidade eletromagnética, desempenho, confiabilidade, projeto e desenvolvimento, segurança e meio ambiente, buscando através do estabelecimento de normas, eliminar barreiras técnicas e, desta forma, promover o comércio (PRESTES, 2010).

4.6.1 IEC 61000

A avaliação da imunidade eletromagnética é efetuada através das normas da série IEC 61000. Essa série de normas da IEC 61000 aborda tópicos básicos sobre EMC, como terminologias, descrição de fenômenos eletromagnéticos, medições e técnicas de ensaio (IEC, [s. d.]). Esses assuntos encontram-se subdividido em padrões e relatórios técnicos, compostos de partes:

- **Parte 1:** Considerações gerais (introdução, princípios fundamentais, segurança), definições e terminologia;
- **Parte 2:** Descrição e classificação do ambiente e níveis de compatibilidade;
- **Parte 3:** Limites de emissão e imunidade;
- **Parte 4:** Técnicas de medição e ensaio;
- **Parte 5:** Guias de instalação, métodos de mitigação e dispositivos;
- **Parte 6:** Padrões genéricos;
- **Parte 9:** Diversos.

Para esta dissertação foram utilizadas as normas presentes na parte 4, IEC 61000-4-2, IEC 61000-4-4, IEC 61000-4-5 e IEC 61000-4-29, detalhadas a seguir.

4.6.2 IEC 61000-4-2

Esta norma se refere aos requisitos de imunidade e métodos de testes para sistemas sujeitos a descargas de eletricidade estática, tanto para contato direto dos operadores quando a objetos adjacentes. Além disso, define intervalos de níveis de teste que se relacionem com diferentes condições ambientais e de instalações. O objetivo desta norma é estabelecer uma base comum e reproduzível para avaliar o desempenho de sistemas quando submetidos a descargas eletrostáticas que podem ocorrer diretamente por uma pessoa ou por objetos próximos (IEC, 2001). A estrutura definida na norma para este ensaio é a seguinte:

- Forma de onda típica da corrente de descarga;
- Gama de níveis de teste;
- Equipamentos de teste;
- Configuração de teste;

- Procedimento de teste.

A Tabela 2 apresenta os níveis de testes a norma IEC 61000-4-2:

Tabela 2 – Níveis de teste da IEC 61000-4-2.

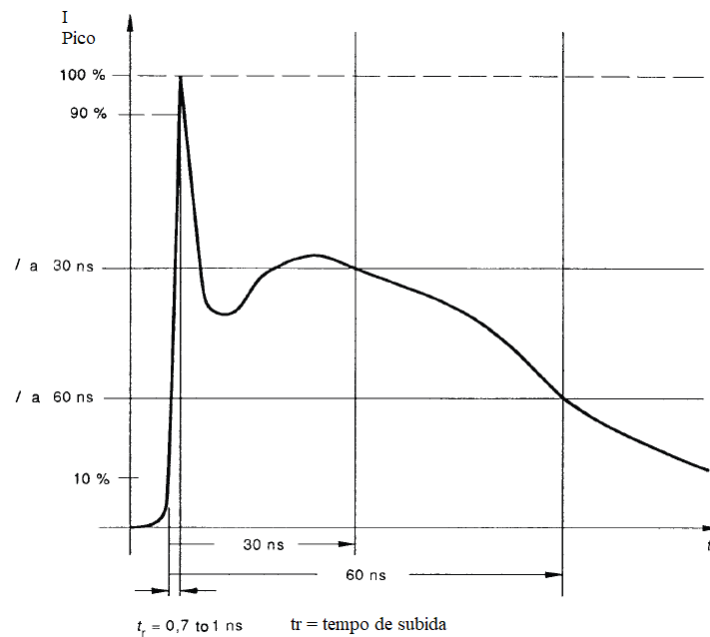
Descarga por Contato		Descarga por Ar	
Nível	Tensão de teste (kV)	Nível	Tensão de teste (kV)
1	2	1	2
2	4	2	4
3	6	3	8
4	8	4	15
x	Especial	x	Especial

"x" é um nível aberto. O nível deve ser especificado na especificação do equipamento dedicado. Pode ser necessário equipamento de teste especial.

Fonte: Adaptado de (IEC, 2001).

Um exemplo da forma de onda especificado pela norma pode ser visto na Figura 20.

Figura 20 - Forma de onda proposta pela IEC 61000-4-2



Fonte: Adaptado de (IEC, 2001).

4.6.3 IEC 61000-4-4

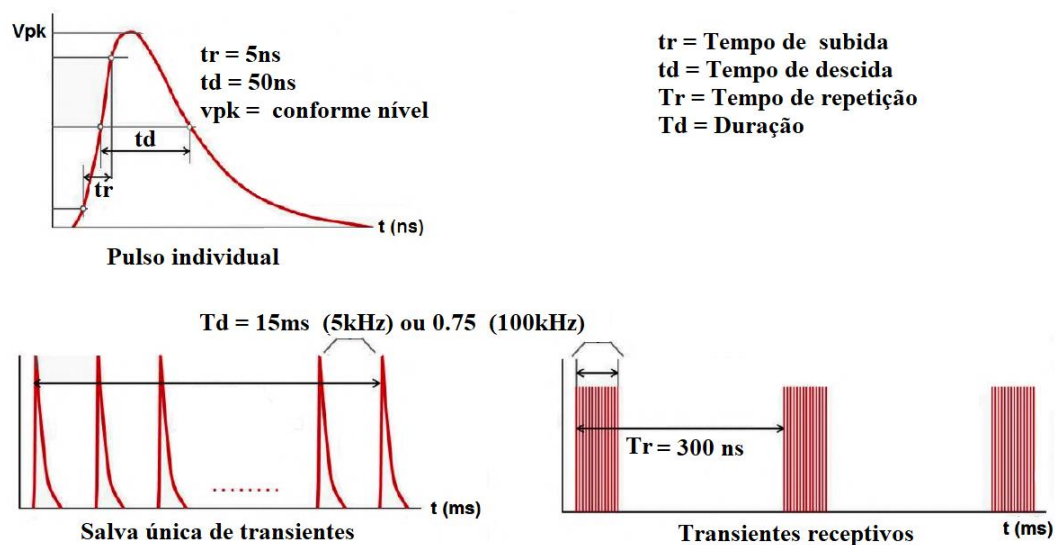
A norma IEC 61000-4-4 se refere a imunidade de equipamentos elétricos e eletrônicos para transientes elétricos rápidos e repetitivos. Fornece requisitos de imunidade e procedimentos de teste relacionados a transientes/picos elétricos rápidos. O objetivo desta norma é estabelecer uma referência comum e reproduzível para avaliar a imunidade de sistemas quando submetidos a EFT nas portas de alimentação, comunicação, controle e aterramento (IEC, 2004).

A estrutura geral do procedimento de ensaio, define os seguintes parâmetros (IEC, 2004):

- Formas de onda da tensão de ensaio;
- Níveis de ensaio;
- Equipamentos, procedimentos de calibração e verificação;
- Configuração e procedimento de ensaio.

Uma alta tensão de ensaio com alta amplitude, curto tempo de subida, alta frequência de repetição e baixa energia dos transientes, são aspectos importantes para a execução do teste. A Figura 21 apresenta as formas de ondas definidas pela IEC 61000-4-4:

Figura 21 - Formas de onda de transientes elétricos rápidos



Fonte: Adaptado de (IEC, 2004)

As especificações do teste são indicadas na Tabela 3.

Tabela 3 - Níveis de teste da norma IEC 61000-4-4.

Nível	Portas de Alimentação e aterramento		Portas de Sinal e Controle	
	Tensão de Pico (kV)	Frequência (kHz)	Tensão de Pico (kV)	Frequência (kHz)
1	0,5	5 ou 100	0,25	5 ou 100
2	1	5 ou 100	0,5	5 ou 100
3	2	5 ou 100	1	5 ou 100
4	4	5 ou 100	2	5 ou 100
Especial	Especial	Especial	Especial	Especial

Fonte: Adaptado de (IEC, 2004).

4.6.4 IEC 61000-4-5

Esta parte da IEC 61000 está relacionada com os requisitos de imunidade, métodos de ensaio e níveis de ensaio recomendados para equipamentos, com relação a surtos unidirecionais causados por sobretensões transitórias provenientes de manobras e descargas atmosféricas. São definidos vários níveis de ensaio, que se relacionam a diferentes condições de ambiente e instalação. Estes requisitos foram desenvolvidos e são aplicáveis aos equipamentos elétricos e eletrônicos. O objetivo desta norma é estabelecer uma referência comum para avaliar a imunidade de equipamentos elétricos e eletrônicos quando sujeitos a sobretensões. O método de teste documentado nesta parte da IEC 61000 descreve um método consistente para avaliar a imunidade de um equipamento ou sistema contra um fenômeno definido (IEC, 2005). A estrutura definida na norma para este ensaio é a seguinte:

- Gama de níveis de teste;
- Equipamentos de teste;
- Configuração de teste;
- Procedimento de teste.

Tabela 4 - Níveis de teste da norma IEC 61000-4-5 para curto circuito.

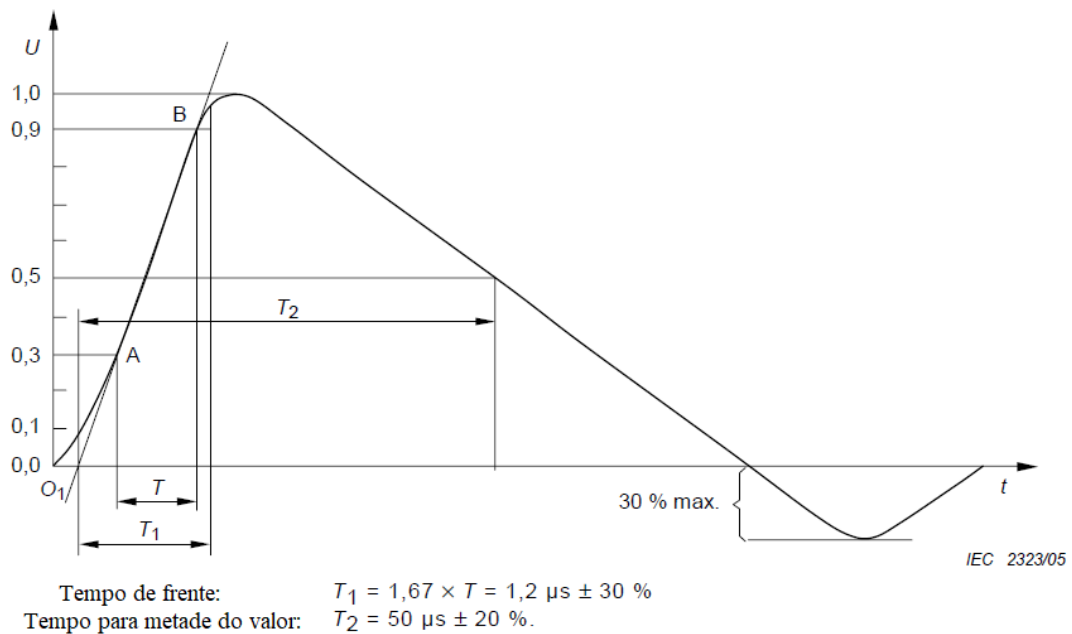
Nível	Tensão de teste de circuito aberto $\pm 10\%$ (kV)	Nível	Corrente de pico de curto-circuito $\pm 10\%$ (kA)
1	0,5	1	0,25
2	1	2	0,5
3	2	3	1
4	4	4	2
x	Especial	x	Especial

"x" é um nível aberto. Pode ser qualquer nível, acima, abaixo ou entre os outros níveis. Pode ser especificado no padrão do produto

Fonte: Adaptado de (IEC, 2005).

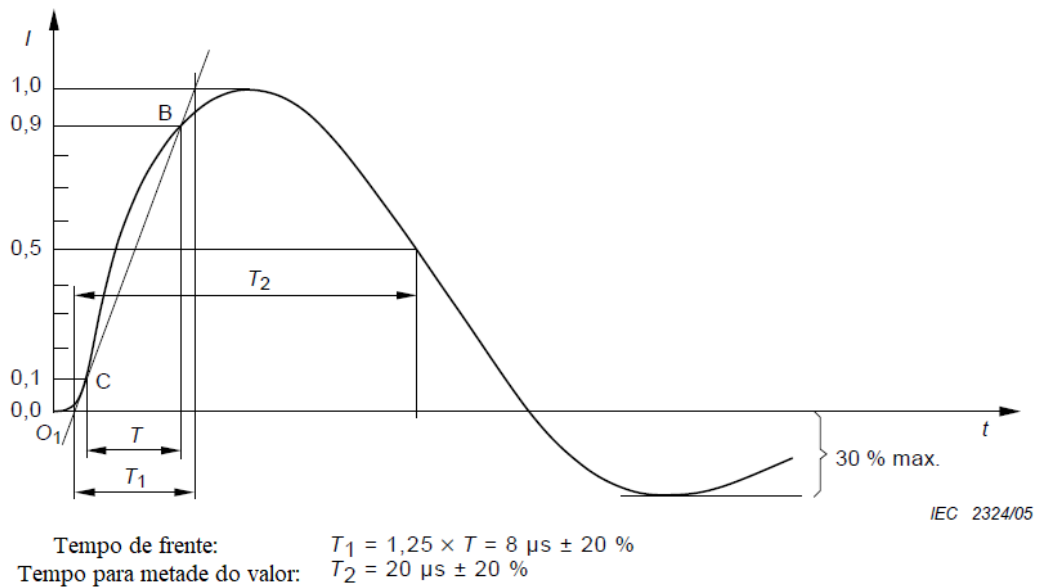
Um exemplo da forma de onda de tensão de circuito aberto pode ser visto na Figura 22, bem com a forma de onda corrente de curto circuito, pode ser vista na Figura 23.

Figura 22 - Forma de onda de tensão de circuito aberto IEC 61000-4-5.



Fonte: Adaptado de (IEC, 2005).

Figura 23 - Forma de onda de corrente de curto circuito IEC 61000-4-5.



Fonte: Adaptado de (IEC, 2005).

4.6.5 IEC 61000-4-29

Esta parte da IEC 61000 define métodos de teste para imunidade a quedas de tensão, interrupções e variações nas portas de alimentação de dispositivos de corrente contínua. O objetivo desta norma é estabelecer uma base comum e reproduzível para teste em equipamentos elétricos e eletrônicos quando sujeitos a quedas de tensão, interrupções curtas ou variações de tensão em corrente contínua (IEC, 2000).

Tendo em vista a existência de diversos tipos de distúrbios relacionados às linhas de alimentação de equipamentos e dispositivos elétricos e eletrônicos, queda de tensão, pequena interrupção e variação de tensão podem ser definidos (IEC, 2000):

- Queda de tensão:** é caracterizada como uma súbita redução na tensão de alimentação, do equipamento e/ou dispositivo, seguida da sua recuperação em um curto período de tempo.
- Pequena Interrupção:** é caracterizada como o desaparecimento momentâneo da tensão por um período de tempo não maior que um minuto. Na prática, quedas de tensão superiores a 80% são consideradas interrupções.

- c) **Variação de tensão:** é caracterizada como uma mudança gradual da tensão de alimentação para um valor maior ou menor do que a tensão nominal (U_T), podendo ser a sua duração curta ou longa.

As tabelas abaixo apresentam os níveis de tensão percentuais relativos à tensão nominal (U_T) e os tempos de duração sugeridos pela norma IEC 61000-4-29 para testes em equipamentos e dispositivos elétricos e eletrônicos, com quedas de tensão, pequenas interrupções e variações de tensão.

Tabela 5 - Níveis de tensão recomendados para teste de queda de tensão.

Teste	Nível de Tensão (% UT)	Duração (s)
		0,01
		0,03
Queda de Tensão	40 a 70 ou x	0,1
		0,3
		1
		x
		x: valor definido de acordo com a especificação do produto

Fonte: Adaptado de (IEC, 2000).

Tabela 6 - Níveis de tensão recomendados para teste de pequena interrupção de tensão.

Teste	Nível de Tensão (% UT)	Duração (s)
		0,001
		0,003
		0,01
Pequena Interrupção	0	0,03
		0,1
		0,3
		1
		x
x: valor definido de acordo com a especificação do produto		

Fonte: Adaptado de (IEC, 2000).

Tabela 7 - Níveis de tensão recomendados para teste de variação de tensão.

Teste	Nível de Tensão (% UT)	Duração (s)
Variação de Tensão	85 a 120	0,01
		0,03
	ou	0,1
	80 a 120	0,3
	ou	1
	x	x
x: valor definido de acordo com a especificação do produto		

Fonte: Adaptado de (IEC, 2000).

Parte II

Metodologia

5 SIMULAÇÃO DAS FORMAS DE ONDA E BASE DE DADOS

Neste capítulo são apresentados os dados referentes a simulação das formas de onda propostas pelas normas citadas nos capítulos anteriores, bem como a criação do conjunto de dados utilizados na criação e treinamento da rede neural.

O computador utilizado para a execução do treinamento e classificação da RNA foi um notebook com processador Intel Core i5-7200 com 2.5Ghz, 16Gb de memória RAM e sistema operacional Windows 10.

5.1 Introdução

No capítulo anterior foram apresentadas normas e regulamentações que dizem respeito a EMC e EMI, destacando as quatro normas que embasam essa dissertação. As normas técnicas trazem artefatos para o procedimento dos testes bem com a apresentação dos circuitos relacionados a cada teste para obtenção das formas de ondas citadas, entretanto para este trabalho foi definido que as formas de ondas propostas seriam oriundas de simulações, visto que, o algoritmo da RNA também foi desenvolvido a nível de simulação e não foi embarcado em nenhum dispositivo eletrônico. A criação das formas de onda através de simulação agrega para a construção de um *Dataset*, representando as formas de ondas das normas IEC 61000-4-2, IEC 61000-4-4, IEC 61000-4-5 e IEC 61000-4-29 a fim de contribuir com trabalhos futuros e validar a proposta de criar novas metodologias de testes para EMC e EMI.

5.2 Ferramenta Computacional para Simulação das Formas de Onda

Para a simulação das formas de ondas das normas citadas, a ferramenta escolhida foi o *software* MATLAB (*Matrix Laboratory*), o qual é um *software* que possui uma linguagem de programação para computação técnica e científica, tendo sua primeira versão, escrita na Universidade do Novo México e Stanford no final da década de 70, sendo destinado a cursos de teoria matricial, álgebra linear e análise numérica (SANTANA, 1998).

O MATLAB apresenta alta popularidade no meio acadêmico e científico, pela facilidade de criar programas em relação a outras linguagens de programação. Por conta disso, inúmeros trabalhos científicos são realizados utilizando este software (FILHO, 2009).

5.3 Implementação do algoritmo no MATLAB

As tabelas de níveis de testes fornecidas pelas normas pertencentes ao escopo mencionado anteriormente e apresentadas no referencial teórico deste trabalho, foram a base da entrada de dados para início do algoritmo. Devido aos curtos intervalos de tempos contidos nas especificações das tabelas, o tamanho da amostra no vetor de tempo criado para obtenção das formas de onda foi crucial, um passo muito alto no eixo de tempo, não teria precisão suficiente para representar as formas de ondas. Por exemplo, para representar uma forma de onda de uma pequena interrupção com duração de 0,001 segundos e criarmos uma forma de onda de duração de 5 segundos, seriam necessários 5000 valores de tensão no eixo x. Sendo assim, as formas de onda foram representadas com duração de no máximo 2 segundos e os níveis de intervalos de tempos adotados foram de 0,01 segundos.

5.3.1 Simulação da forma de onda da IEC 61000-4-2

Algumas literaturas apresentam equações para descrever a forma de onda da corrente de uma ESD (TURITSYNA; WEBB, 2005; WANG *et al.*, 2003), porém na versão de 2008 da IEC 61000-4-2, foi incluída uma fórmula analítica para a corrente ESD e foi comprovado que esta é a equação mais precisa para estudar a corrente ESD e é dado pela equação (KATSIVELIS *et al.*, 2010):

$$i(t) = i_1(t) + i_2(t) \quad (5.1)$$

Onde:

$$i_1(t) = \frac{I_1}{e^{\left[\frac{\tau_1}{\tau_2} \left(\frac{n\tau_2}{\tau_1}\right)\right]^{1/n}}} \cdot \frac{\left(\frac{t}{\tau_1}\right)^n}{1 + \left(\frac{t}{\tau_1}\right)^n} \cdot e^{-\left(\frac{t}{\tau_2}\right)} \quad (5.2)$$

$$i_2(t) = \frac{I_2}{e^{\left[\frac{-\tau_3 \cdot (n\tau_4)}{\tau_4}\right]^{1/n}}} \cdot \frac{\left(\frac{t}{\tau_3}\right)^n}{1 + \left(\frac{t}{\tau_3}\right)^n} \cdot e^{-\left(\frac{t}{\tau_4}\right)} \quad (5.3)$$

Examinando a corrente ESD de uma descarga sob uma tensão de carga de 4kV, os valores dos parâmetros obtidos são: $I_1 = 16.6A$, $I_2 = 9.3A$, $\tau_1 = 1.1ns$, $\tau_2 = 2ns$, $\tau_3 = 12ns$, $\tau_4 = 37ns$ e $n = 1.8$ (IEC, 2008).

De posse das equações e dos valores de parâmetros obtidos para uma tensão de 4kV, pode-se utilizar os mesmos parâmetros para os demais níveis de tensão contidos na tabela de níveis de teste da IEC 61000-4-2, sendo necessário apenas uma multiplicação de ganhos na equação 5.1, sendo os valores de ganho 0,5 para uma tensão de 2kV, 1,5 para uma tensão de 6kV e um ganho de 2 vezes para uma tensão de 8kV, valores de tensão definidos para descarga por contato na Tabela 2. Essas equações foram transcritas para o MATLAB, sendo criada uma função que recebe apenas o valor do ganho e devolve a forma de onda para os respectivos níveis solicitados de teste.

A norma define uma série de características para a forma de onda, como valores de pico de corrente em determinado tempo e valores da corrente ao passar do tempo, essas características são vistas na Tabela 8.

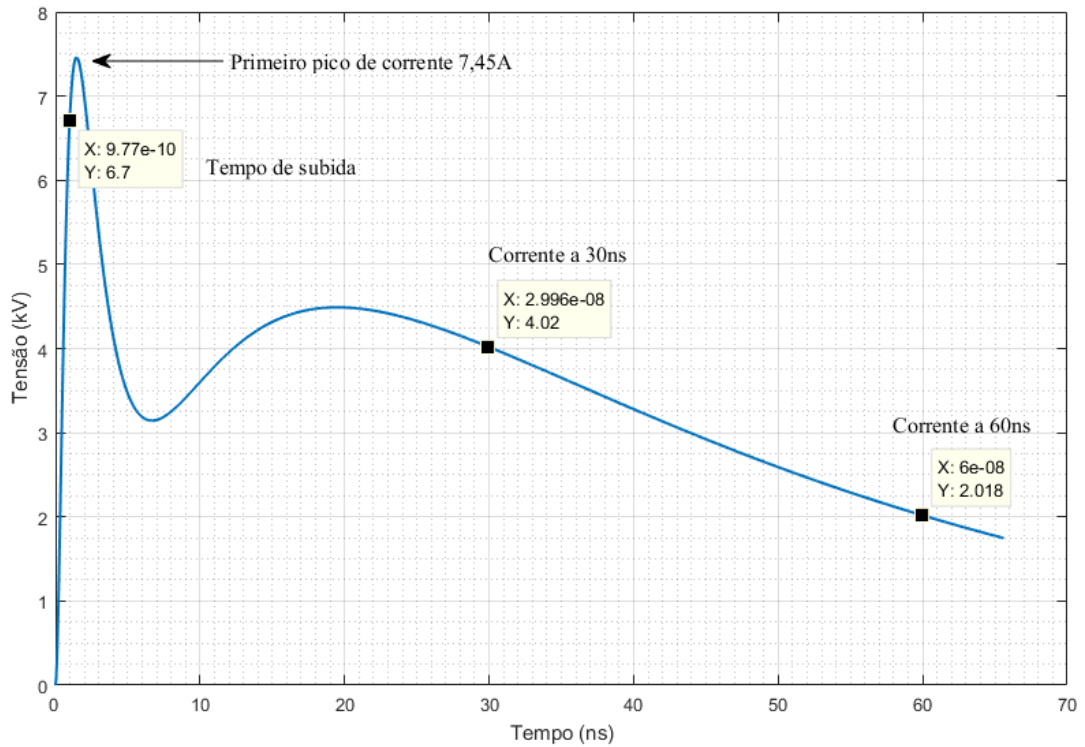
Tabela 8 - Parâmetros da forma de onda da corrente de ESD

Nível	Tensão Indicada	Primeiro pico de corrente ESD $\pm 10\%$	Tempo de subida tr	Corrente ($\pm 30\%$) aos 30ns	Corrente ($\pm 30\%$) aos 60ns
	kV	A	ns	A	A
1	2	7,5	0,7 a 1	4	2
2	4	15	0,7 a 1	8	4
3	6	22,5	0,7 a 1	12	6
4	8	30	0,7 a 1	16	8

Fonte: Adaptado de (IEC, 2001).

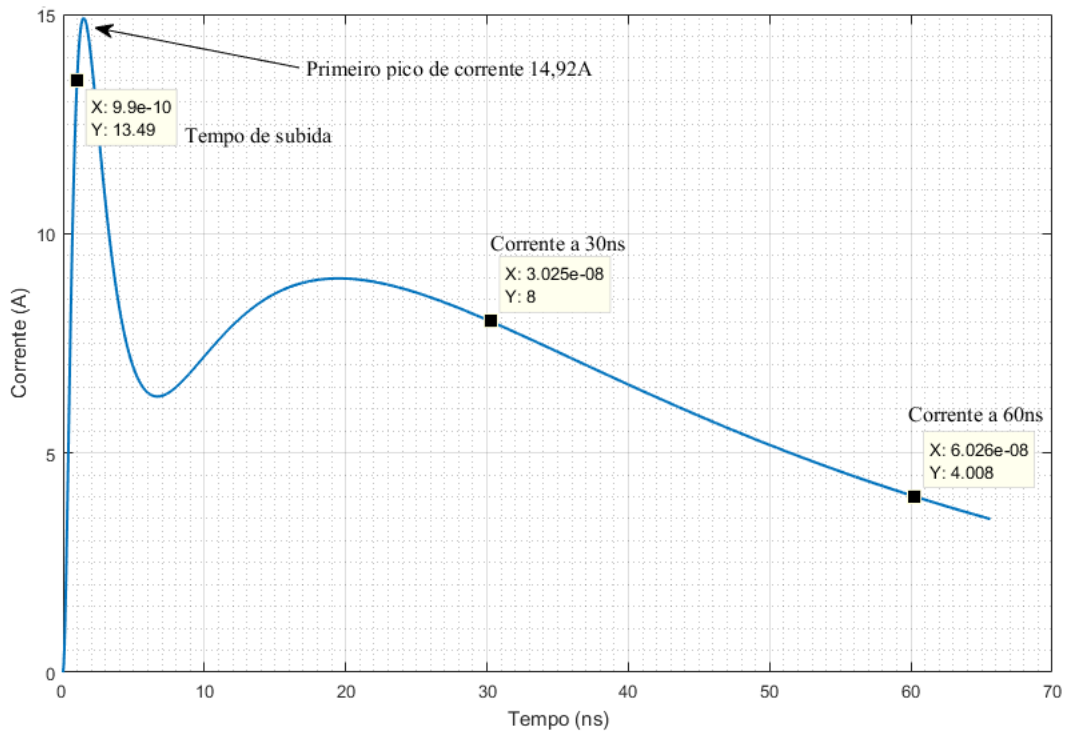
Com base na tabela acima, podemos validar as formas de ondas geradas no MATLAB para os testes de ESD. As formas de onda geradas para os testes mencionados na Tabela 2 por descarga de contato podem ser vistas nas Figuras abaixo:

Figura 24 - Forma de onda da corrente para teste ESD de tensão de 2kV.



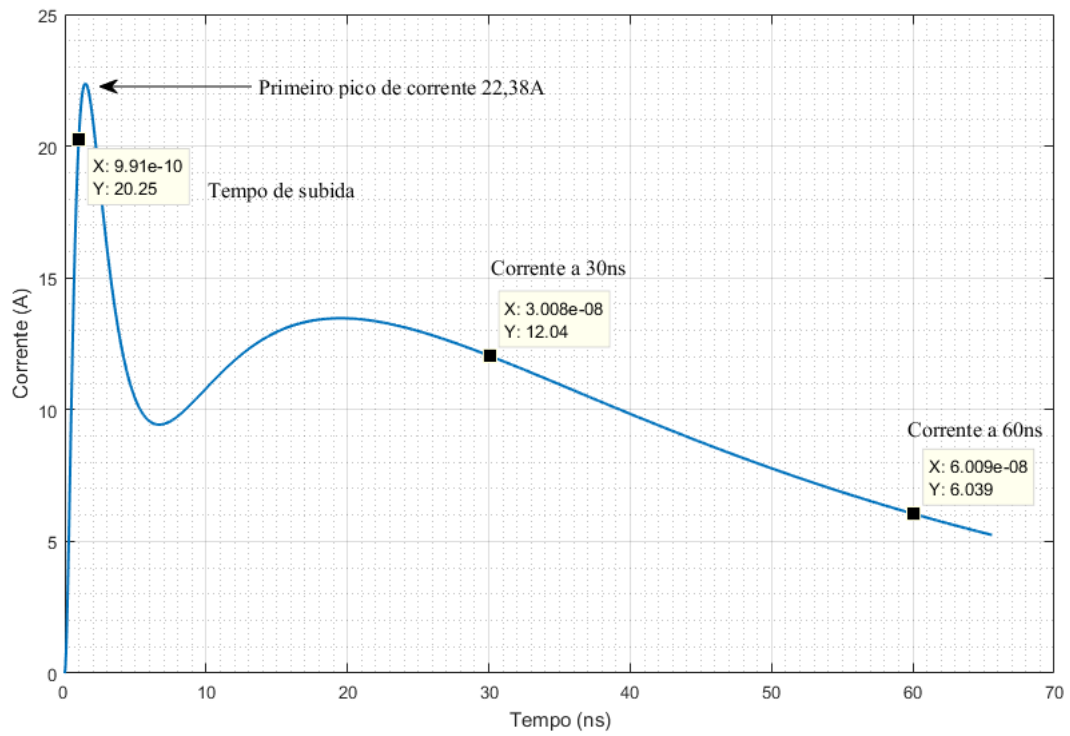
Fonte: Autor.

Figura 25 - Forma de onda da corrente para teste ESD de tensão de 4kV.



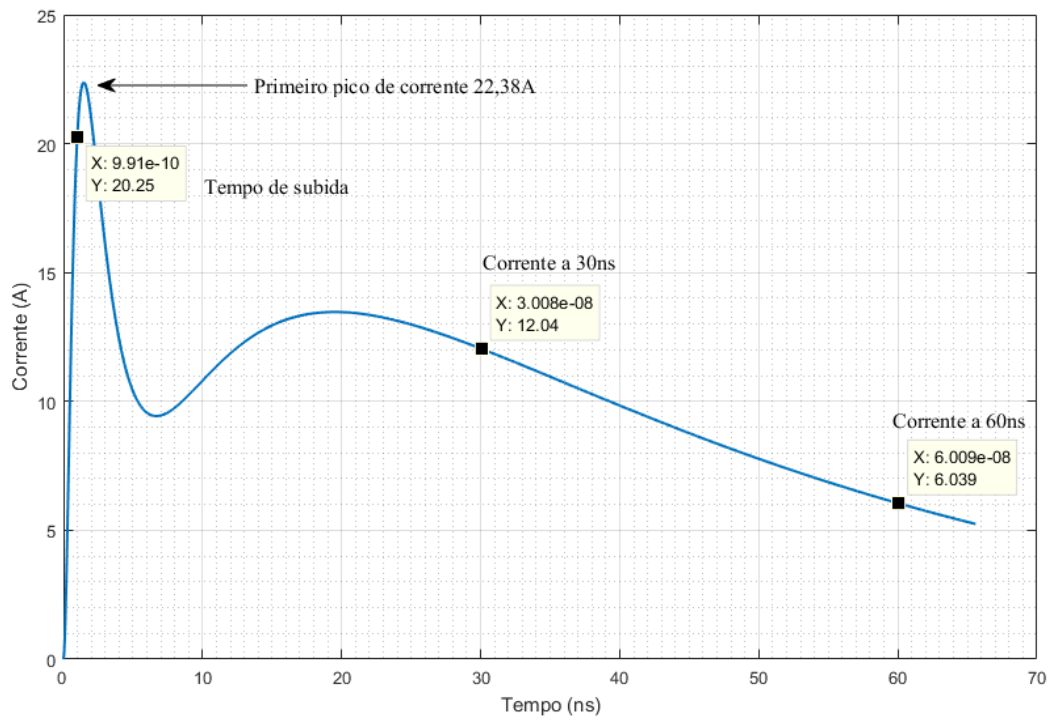
Fonte: Autor.

Figura 26 - Forma de onda da corrente para teste ESD de tensão de 6kV.



Fonte: Autor.

Figura 27 - Forma de onda da corrente para teste ESD de tensão de 8kV.

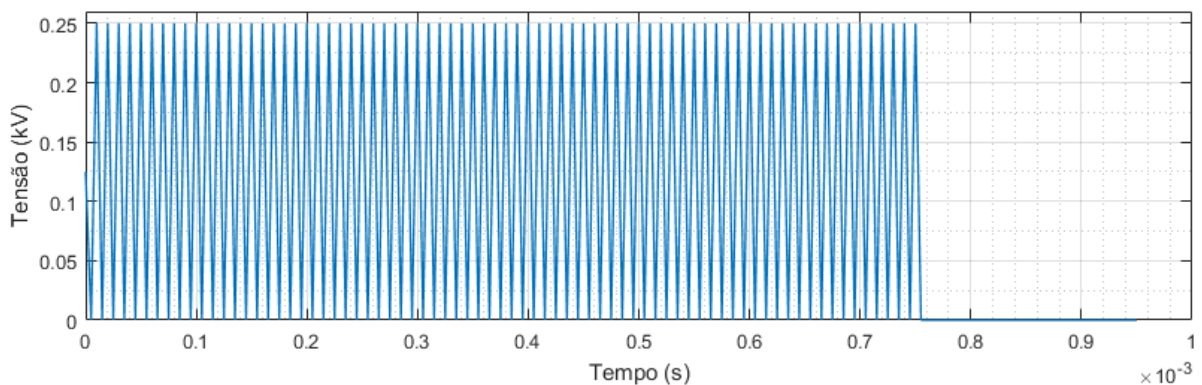


Fonte: Autor.

5.3.2 Simulação da forma de onda da IEC 61000-4-4

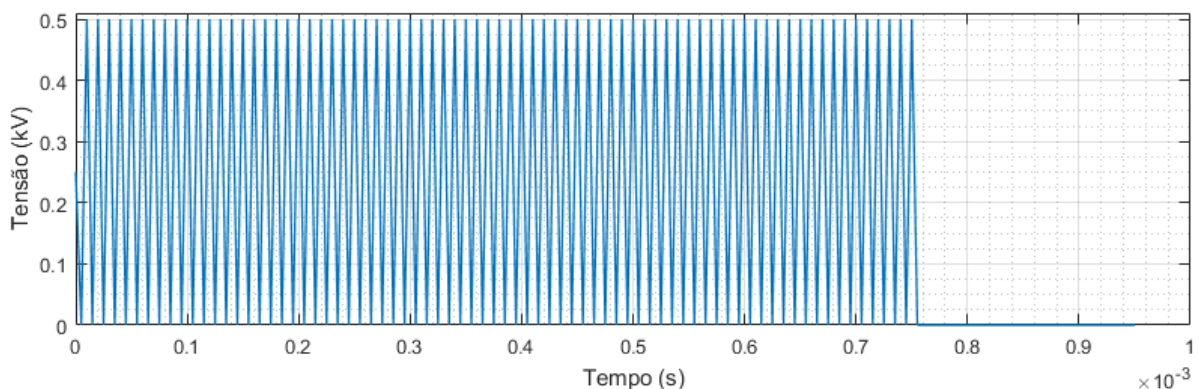
Para gerar as formas de onda, para esta norma, foi utilizada a função *heaviside* do MATLAB, conhecida como função degrau unitário, onde é nula para valores de x menor que zero e 1 para quando x possui valores maiores que zero. Desta forma, podemos multiplicar os valores de picos de tensão definidos pela Tabela 3, pela função de *heaviside* e levar em conta os parâmetros como taxa de duração e repetição, informados pela norma para gerar a forma de onda. Para essa simulação foram utilizados os níveis de teste definidos para os sinais de *I/O* (em inglês, *input/output*) e portas de controle com uma taxa de repetição de 100kHz e uma duração de 75ms. As formas de onda simuladas, geradas pelo MATLAB, podem ser vistas nas Figuras abaixo:

Figura 28 - Formas de onda para teste de EFT com picos de 0,25kV.



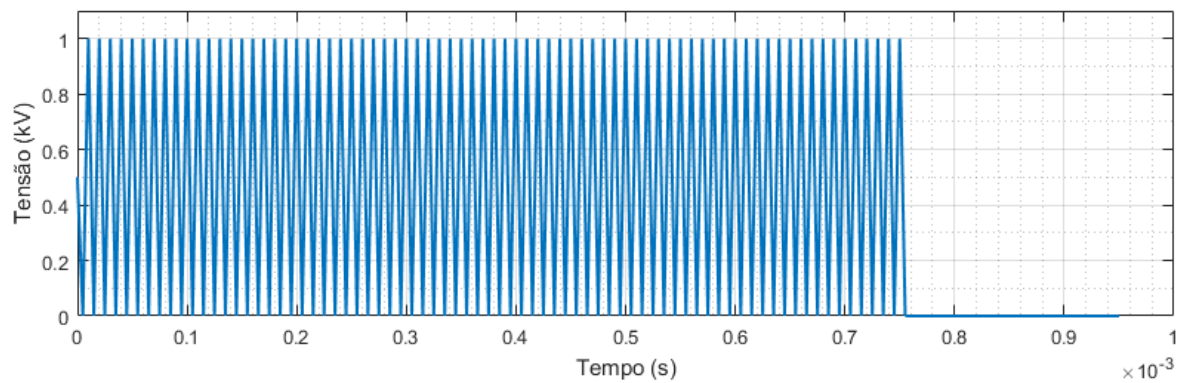
Fonte: Autor.

Figura 29 - Formas de onda para teste de EFT com picos de 0,5kV.



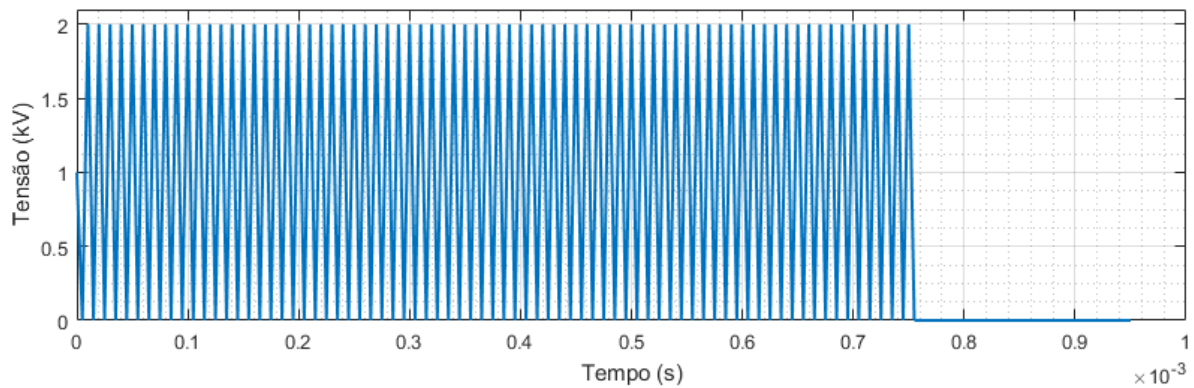
Fonte: Autor.

Figura 30 - Formas de onda para teste de EFT com picos de 1kV.



Fonte: Autor.

Figura 31 - Formas de onda para teste de EFT com picos de 2kV.



Fonte: Autor.

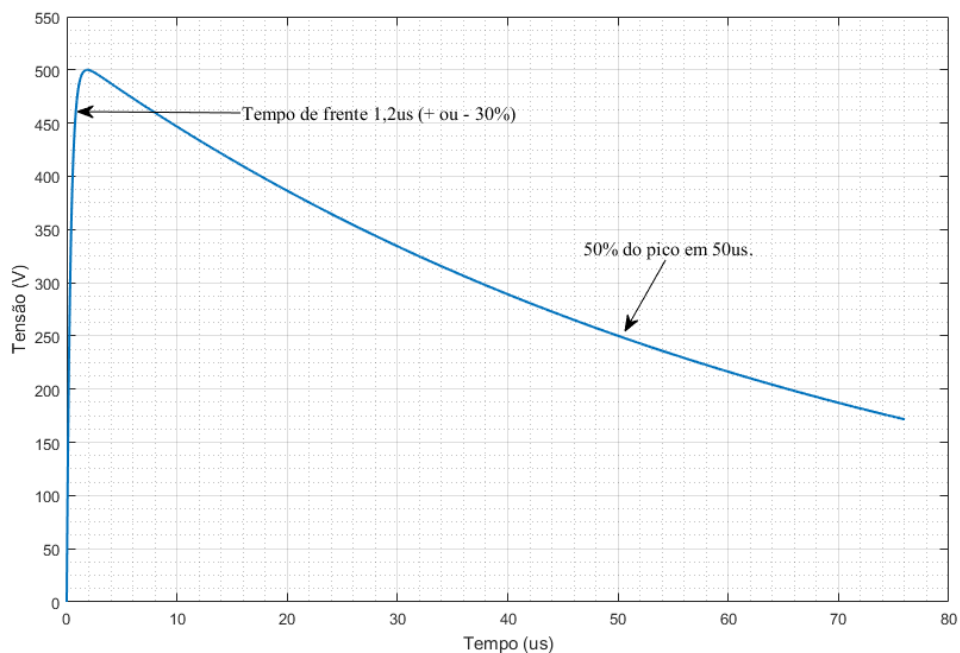
5.3.3 Simulação da forma de onda da IEC 61000-4-5

Na literatura são apresentadas mais de uma maneira para representar as formas de onda dos surtos eletromagnéticos, que são usadas para testes de imunidade e suportabilidade de equipamentos a surtos (BORSOI, 2018). A equação descrita a seguir pode ser utilizada para implementação de uma função para extrair a forma de onda padrão sugerida para testes de surto, e pode ser escrita em termos de duas exponenciais (PSCAD, [s. d.] apud IEC, 2005; IEEE, 2003):

$$x(t) = k(e^{-at} - e^{-\beta t}) \quad (5.4)$$

Onde k , α , β são constantes e t representa o tempo. A equação 5.4 foi utilizada para gerar a forma de onda da tensão de circuito aberto. Para gerar a forma de onda da tensão de circuito aberto, os valores das constantes da equação 5.4 foram utilizadas conforme (PSCAD, [s. d.]), além disso, foi multiplicado o valor de amplitude da tensão correspondente ao nível do teste fornecido pela Tabela 4. Para criação desta função os valores das constantes foram definidos como: $k = 1,0328$, $\alpha = 0,0145$ e $\beta = 2,835$. Para simulação desta forma de onda, ainda foram observados o tempo de frente, que é um parâmetro virtual entre os instantes que o impulso é de 30% a 90% do valor de pico, e também o tempo para o valor do impulso atingir metade do valor de pico. A Figura 32 apresenta um exemplo de forma de onda gerada pela simulação do MATLAB para a tensão de circuito aberto de $1,2 \mu s \pm 30\%$ para o tempo de frente, $50 \mu s \pm 20\%$ para atingir metade do valor e uma tensão de pico de 500V.

Figura 32 - Forma de onda para teste tensão de circuito aberto da IEC 61000-4-5.



Fonte: Autor.

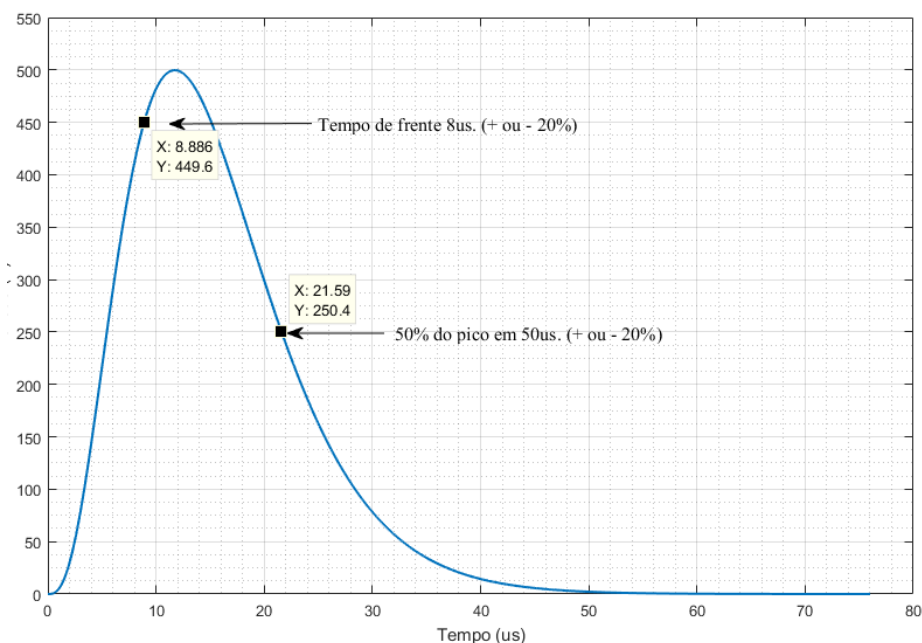
Para gerar a forma de onda da corrente de curto circuito, foi utilizada a seguinte equação (BORSOI, 2018):

$$x(t) = A \times I_p \times t^3 \times e^{\left(\frac{-t}{\tau}\right)} \quad (5.5)$$

Onde: $\tau = 3,911$, $A = 0,01243$, I_p = Valor de pico da corrente de curto circuito e t = tempo.

Com base na equação 5.5, podemos obter as formas de onda da corrente de curto circuito de $8 \mu s \pm 20\%$ de tempo de frente e $20 \mu s \pm 20\%$ para atingir metade do valor de pico de corrente, a forma de onda da simulação implementada no MATLAB, para um pico de corrente de 500A pode ser vista na Figura 33.

Figura 33 - Forma de onda para teste corrente de curto circuito da IEC 61000-4-5.

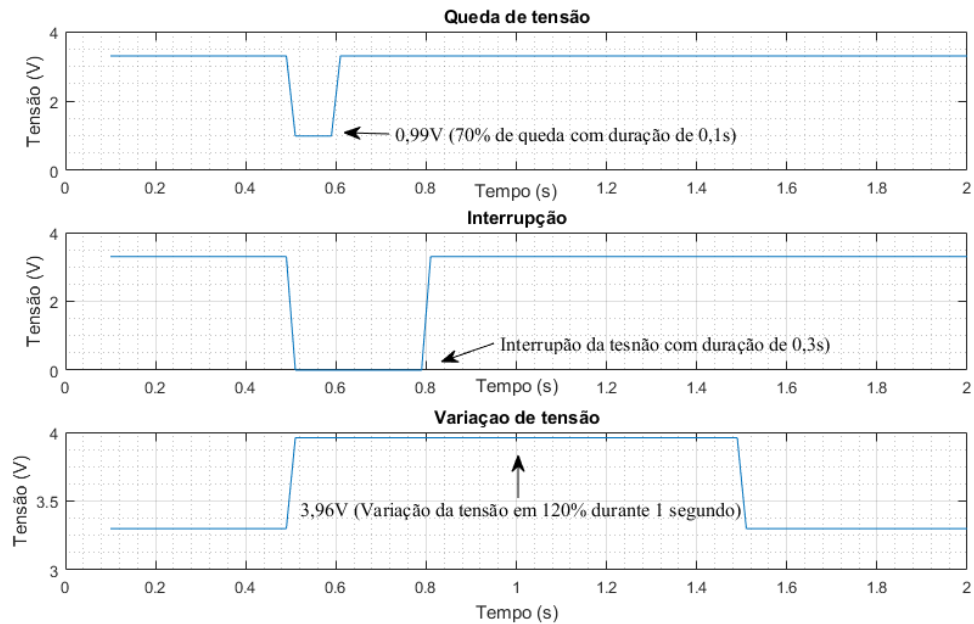


Fonte: Autor.

5.3.4 Simulação da forma de onda da IEC 61000-4-29

Para gerar as formas de onda, para esta norma, também foi utilizada a função *heaviside* do MATLAB, criando uma composição de degraus unitários obedecendo as características fornecidas pela norma. Foi criada uma função desenvolvida no algoritmo que é capaz de receber os parâmetros indicando se é uma queda de tensão, uma pequena interrupção ou variação, bem como, os níveis do percentual de queda e variação, e também os tempos de duração para cada evento. Exemplos das formas de onda geradas para os seguintes testes: Queda de tensão de 70% do valor nominal (3,3V) com duração de 0,1s, pequena interrupção da tensão nominal durante 0,3s e um variação de tensão de 120% durante 1s, podem ser vistas na Figura 34.

Figura 34 - Exemplos simulações geradas para testes descritos na IEC 61000-4-5.

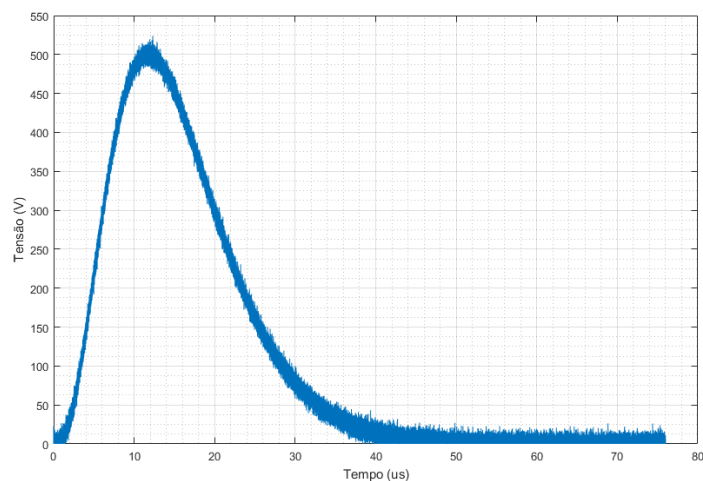


Fonte: Autor.

5.3.5 Inserção de ruído nas formas de onda simuladas

Para tornar os dados mais significativos e não cobrir apenas situações normais citadas pelas normas, visando um comportamento que poderia acontecer em um dispositivo real, foi inserido um ruído branco gaussiano em toda a extensão da forma de onda. Tendo assim, valores discriminados pelas normas para representar a forma de onda e também a forma de onda com valores ruidosos inseridos em seu meio. A Figura 35 apresenta um exemplo da forma de onda apresentada na Figura 33, porém com o ruído inserido.

Figura 35 - Forma de onda para teste corrente de curto circuito da IEC 61000-4-5 com ruído inserido



Fonte: Autor.

5.4 Criação da base de dados

O algoritmo desenvolvido no MATLAB é capaz de gerar todas as formas de onda simuladas ao mesmo tempo, alocando os seus valores de x e y em vetores. Como dito no capítulo sobre a implementação do algoritmo, devido aos curtos intervalos de tempos definidos pelas tabelas das normas, a representação de uma forma de onda em vetor, ficou contendo 191 posições de valores que representam os valores de tensão ou corrente ao longo do tempo.

Para definir a qual forma de onda o vetor pertence, ao final do mesmo, foi acrescentada uma nova posição para definir sua classe. Desta forma um vetor capaz de representar uma forma de onda especificada por alguma das normas, ficou contendo 192 posições para representá-la. A Tabela 9 apresenta as classes criadas durante a simulação e o valor de sua representação a ser inserido na última posição de cada vetor, visando sua futura classificação.

Tabela 9 - Normas e suas representações na base de dados

Norma	Classe	Representação
IEC 61000-4-29	Quedas de tensão	1
	Pequenas interrupções	2
	Variações de tensão	3
IEC 61000-4-4	Transientes rápidos	4
IEC 61000-4-5	Surto tensão aberta	5
	Surto curto circuito	6
IEC 61000-4-2	Descarga eletrostática	7
x	Tensão contínua	8

Fonte: Autor.

Nota-se ainda, que além das formas de onda para as normas estabelecidas no escopo teste trabalho, foi acrescentada uma classe chamada de Tensão contínua. Essa classe representa um comportamento além dos citados pelas normas, com o intuito de representar o funcionamento normal e operacional de dispositivo, semelhante ao motivo da inserção do ruído branco.

A fim de ter um maior volume na base de dados, para agregar mais dados para o treinamento da rede neural, os dados dos vetores de cada classe foram replicados, para isso foi criado uma matriz de células para alocar esses vetores. Foi definida uma matriz $1 \times n$, onde n

representa o número de vezes que o vetor para cada representação será repetido. A Figura 36 mostra a alocação das variáveis criadas para cada vetor, o valor de n atribuído e a classe o qual cada vetor representa.

Figura 36 - Variáveis criadas para representação dos vetores e suas respectivas classes.

Nome definido a variável do vetor	1 x n	Representação	Nome definido a variável do vetor	1 x n	Representação
v voltageDipsRuido	1x36 cell	1	v sourceOpenRuido	1x24 cell	5
v voltageDips	1x36 cell		v sourceOpen	1x24 cell	
v shortInterruptionRuido	1x24 cell	2	v sourceShortRuido	1x24 cell	6
v shortInterruption	1x24 cell		v sourceShort	1x24 cell	
v voltageVariationRuido	1x36 cell	3	v dischargeRuido	1x24 cell	7
v voltageVariation	1x36 cell		v discharge	1x24 cell	
v transientesRuido	1x16 cell	4	v constanteRuido	1x30 cell	8
v transientes	1x16 cell		v constante	1x30 cell	

Fonte: Autor.

Com a redundância criada, ao final dos testes e execuções, foram totalizados 428 vetores para representar as 8 classes definidas na base de dados, descritas na Tabela 9.

Para a exportação da base criada, foram utilizadas funções do MATLAB para escrita e manipulação de arquivos. Desta forma, os vetores criados são escritos em um arquivo do tipo CSV (nomenclatura vinda de *comma Separated Values*, que em português significa, Valores Separados por Vírgula), com um cabeçalho informando número da amostra, o valor da respectiva posição e a classe ao final do vetor como dito anteriormente. Um exemplo do arquivo gerado com a base dados pode ser visto na Figura 37.

Figura 37 - Arquivo CSV exportado com a base de dados

Amostra 1	Amostra 2	Amostra 3	Amostra 4	Amostra 5	Amostra 6	...	Amostra 188	Amostra 189	Amostra 190	Amostra 191	Forma de onda
0	3.300	3.300	3.300	3.300	3.300	3.300	3.300	3.300	3.300	3.300	1.0
1	3.300	3.300	3.300	3.300	3.300	3.300	3.300	3.300	3.300	3.300	1.0
2	3.300	3.300	3.300	3.300	3.300	3.300	3.300	3.300	3.300	3.300	1.0
3	3.300	3.300	3.300	3.300	3.300	3.300	3.300	3.300	3.300	3.300	1.0
4	3.300	3.300	3.300	3.300	3.300	3.300	3.300	3.300	3.300	3.300	1.0
...
423	2.388	2.549	2.543	2.438	2.465	2.414	2.498	2.474	2.484	2.490	8.0
424	3.147	3.149	3.284	3.396	3.323	3.239	3.210	3.120	3.122	3.098	8.0
425	3.488	3.259	3.411	3.336	3.202	3.450	3.443	3.438	3.356	3.451	8.0
426	3.372	3.328	3.281	3.244	3.224	3.290	3.327	3.229	3.274	3.352	8.0
427	3.258	3.406	3.223	3.154	3.346	3.279	3.220	3.297	3.217	3.384	8.0

428 rows x 192 columns

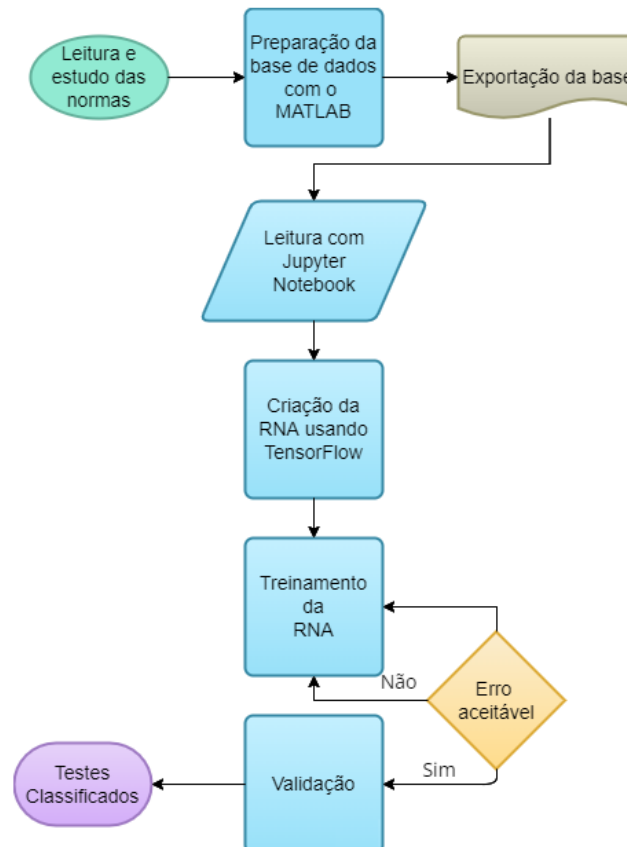
Fonte: Autor.

6 CRIAÇÃO DA RNA E TREINAMENTO

6.1 Introdução

Neste trabalho, utiliza-se redes neurais artificiais do tipo MLP, sendo o treinamento realizado através do método de *backpropation* (conceitos detalhados no Capítulo 2). A estrutura do desenvolvimento seguido até aqui, pode ser visto na Figura 38.

Figura 38 - Fluxo de desenvolvimento



Fonte: Autor.

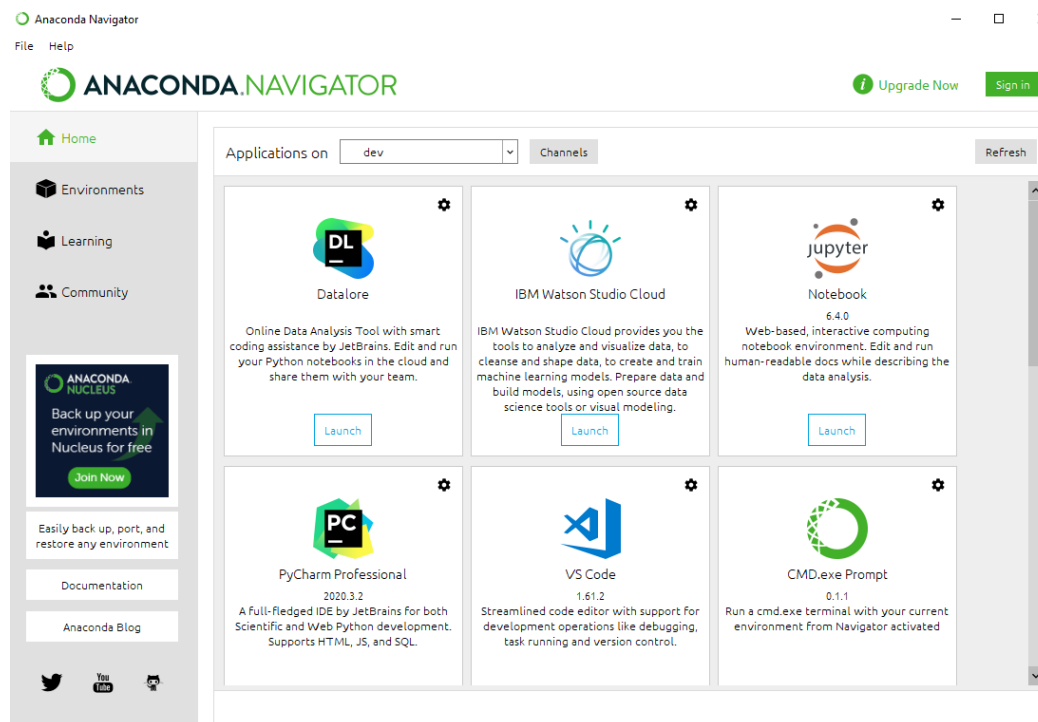
As primeiras três etapas ilustradas no fluxo de desenvolvimento foram discutidas nos capítulos anteriores, a seguir será tratado os demais pontos envolvidos no desenvolvimento para criação e treinamento da RNA.

6.2 Ambiente de desenvolvimento

Ambientes de desenvolvimento se caracterizam por entregarem ferramentas de edição e produtividade que podem aumentar a efetividade na produção e análise de um *software* (DE MEDEIROS, 2021).

O modelo da RNA foi desenvolvido no ambiente do *Anaconda* que é uma das mais populares plataformas para ciência de dados, possuindo em seu ambiente, o *Anaconda Navigator*, que é uma aplicação com interface de usuário, o qual gerencia pacotes e variáveis e entrega diversos recursos para o desenvolvimento de aplicações voltadas a ciências de dados sem a necessidade do uso de linhas de comando (GUIMARÃES, 2019) (ANACONDA, 2021). Para a escrita do algoritmo, importação das bibliotecas e modelagem da RNA foi utilizado o *Jupyter Notebook*, que é uma aplicação *web* de código aberto que permite ao desenvolvedor criar código em diversas linguagens como por exemplo, *Python*, *R*, *Scala*, entre outras. Além de já contar com recursos de *Big Data*. A Figura 39, apresenta a visualização do *software Anaconda Navigator* e algumas de suas ferramentas disponíveis.

Figura 39 - Interface do Anaconda Navigator.



Fonte: Autor.

6.3 Ferramentas utilizadas para criação e treinamento da RNA

Vale destacar que a linguagem de programação adotada nesta etapa da metodologia foi o *Python*, que é uma linguagem interpretada, orientada a objetos, de alto nível e com semântica dinâmica. O *Python* tem uma sintaxe simples e fácil de aprender, que enfatiza a legibilidade, reduzindo o custo de manutenção nos programas. Permite ainda o uso de módulos e pacotes, facilitando a modularidade e reutilização de código (SANTOS, G. C., 2020). As principais bibliotecas utilizadas no contexto do desenvolvimento são citadas na sequência.

6.3.1 *Pandas*

A leitura e carregamento da base dados foi possível por meio da biblioteca *Pandas*. Essa biblioteca é um pacote *Python* que fornece estrutura de dados rápidas, flexíveis e expressivas, projetadas para facilitar o trabalho com dados estruturados (tabulares, multidimensionais, potencialmente heterogêneos) e de séries temporais (PANDAS, 2018).

6.3.2 *Scikit-Learn*

O *Scikit-Learn* é uma das bibliotecas de Aprendizado de Máquina mais conhecidas e utilizadas do *Python*, dentre as diversas existentes, elaborada em código aberto e desenvolvida para suportar e possibilitar o treino de diversas técnicas de estatística e Aprendizado de Máquina, para aprendizagem supervisionada e não supervisionada (ESCOVEDO, 2019). O *Scikit-Learn*, também conhecido como *sklearn*, oferece recursos para diferentes pontos do desenvolvimento de uma RNA. Neste trabalho, a biblioteca *sklearn* foi utilizada para técnicas de preparação da base dados e também para dividir a base de dados em amostras de teste e amostras de treinamento.

6.3.3 *NumPy*

Durante o desenvolvimento algumas operações matemáticas, utilizando vetores e matrizes, foram necessárias, e foram realizadas por meio da biblioteca *NumPy*. Essa biblioteca usa a sintaxe e a semântica do *Python*, operando com matrizes e empregando operações lógicas.

Além disso, permite diversas abordagens orientadas a objetos e operações com tabelas usando vetores (LEMENKOVA, 2019).

6.3.4 *TensorFlow*

O *TensorFlow* é uma biblioteca de código aberto para computação numérica usando grafos computacionais. Originalmente foi desenvolvida pela equipe da *Google Brain Team* para aprendizado de máquina e pesquisa de redes neurais profundas (*Deep Learning*), porém a biblioteca é abrangente o suficiente para ser aplicada a uma grande variedade de outros domínios (MODESTI, 2020).

O *TensorFlow* foi utilizado para as funções necessárias na criação da RNA deste trabalho pois possui implementações previamente disponibilizadas para redes neurais por meio de uma API (do inglês, *Application Programming Interface*) de alto nível que tornam a implementação mais simples, além de ter suporte à computação distribuída (HUANG, 2019).

6.4 Arquitetura da RNA

Nesta seção são apresentadas as etapas realizadas na definição da melhor configuração da RNA para satisfazer a proposta deste trabalho.

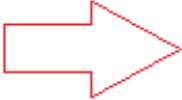
A escolha para este trabalho foi o desenvolvimento e criação de uma Rede MLP *feedforward* utilizando o algoritmo de treinamento *Backpropagation*, porém também faz parte da descrição da arquitetura a determinação da quantidade de neurônios da camada de entrada, quantidade de camadas ocultas e número de neurônios da camada oculta, bem como o número de neurônios da camada de saída.


Na seção 5.4 foi definido que para a representação de uma forma de onda tínhamos um vetor de 192 posições, sendo a última posição a classe o qual o vetor representa. Portanto pode-se definir que a RNA terá 191 neurônios na camada de entrada e 8 neurônios na camada de saída. Quanto ao número de camadas ocultas, foi definida apenas uma camada, que segundo a literatura é suficiente para problemas básicos de classificação (GOODFELLOW; BENGIO; COURVILLE, 2016). O número de neurônios na camada oculta foi definido empiricamente com base em tentativas realizadas durante testes, sendo definidos entre 99 e 350 neurônios. Não existe um método exato para encontrar o número ideal de neurônios (MAS; FLORES, 2008), na seção de resultados é apresentada a exploração da configuração com o número de neurônios na camada oculta e os seus respectivos resultados.

6.5 Funções e parâmetros utilizados na RNA

Após a importação da base de dados e separação dos valores de entrada e saída utilizando o *Pandas*, também foi realizada uma transformação nos valores de saída utilizando o *sklearn*, onde originalmente tínhamos valores de 1 a 8 representando a classe de teste. A transformação é feita para atender a arquitetura definida com os 8 neurônios na camada de saída, essa transformação faz com a saída seja representada na forma de um *array* de 8 posições com valores 0 ou 1, onde nas 8 posições apenas uma posição estará com o valor 1, representando que pertence aquela classe. Um exemplo dos valores de saída antes e depois da transformação podem ser vistos na Figura 40.

Figura 40 - Valor de saída da RNA com transformação

Sem Transformação			Saida transformada
0	1.0		<code>array([[1., 0., 0., ..., 0., 0., 0.],</code> <code>[1., 0., 0., ..., 0., 0., 0.],</code> <code>[1., 0., 0., ..., 0., 0., 0.],</code>
1	1.0		
2	1.0		
3	1.0		
4	1.0		

423	8.0		<code>[0., 0., 0., ..., 0., 0., 1.],</code> <code>[0., 0., 0., ..., 0., 0., 1.],</code> <code>[0., 0., 0., ..., 0., 0., 1.]])</code>
424	8.0		
425	8.0		
426	8.0		
427	8.0		

Fonte: Autor.

A base de dados ainda com a utilização do *sklearn* é dividida entre dados de treinamento e dados de teste, sendo os dados de treinamento aqueles que foram utilizados propriamente na iteração do algoritmo *Backpropagation* e a fatia de testes aquela utilizada para realizar as previsões, a base foi dividida em 50% testes, 50% treinamento. Após a base carregada e preparada e a arquitetura da RNA definida, é implementado o algoritmo que realiza o *feedforward*. Para implementação do *feedforward* é utilizado o *TensorFlow* que permite o uso de funções para a sequência de adição, multiplicação e ativação dos valores alimentados adiante na rede MLP. A

função de ativação utilizada foi a ReLU, o qual suas características foram apresentadas na seção 2.3.3

Para os cálculos de erro a função *softmax_cross_entropy_with_logits_v2* do *TensorFlow* foi utilizada. Essa função mede a probabilidade de erro em tarefas de classificação discreta nas quais as classes são mutuamente exclusivas (cada entrada está em exatamente uma classe).

A função de otimização da RNA utilizada foi a *AdamOptimizer* que é um método estocástico de gradiente descendente (vide Seção 2.6.1) que se baseia na estimativa adaptativa de momentos de primeira e segunda ordem e é computacionalmente eficiente, tem pouca necessidade de memória, invariante ao reescalonamento e é bem adequado para problemas que são grandes em termos de dados/parâmetros (KINGMA; BA, 2015). Para a função do otimizador é definido ainda a taxa de aprendizagem, parâmetro chamado de *learning_rate*, o valor definido para o otimizador foi de 0,0001. A taxa de aprendizado determina em cada iteração do treinamento o valor do tamanho do próximo passo que será dado no gradiente da função de erro.

Outro parâmetro importante definido é o tamanho do lote (do inglês, *Batch Size*), que é um hiperparâmetro (parâmetro que deve ser definido antes de treinar o modelo) que define o número de amostras a serem trabalhadas antes de atualizar os parâmetros do modelo (BROWNLEE, 2018). O tamanho do lote definido para este trabalho foi variado entre 5 e 20, onde o mais adequado é apresentado na seção de resultados.

6.6 Treinamento e previsão da RNA

O método de treinamento utilizado foi o de *Backpropagation*, onde os valores dos pesos e também do bias são alterados a cada iteração a fim de reduzir o erro propagado para a saída da RNA.

Através de um laço de iteração o qual tem sua duração definida pelo parâmetro chamado número de épocas, que é um hiperparâmetro que define o número de vezes que o algoritmo de aprendizado rodara para todo o conjunto de dados de treinamento. Uma época significa que cada amostra no conjunto de dados de treinamento teve a oportunidade de atualizar os parâmetros internos do modelo. Uma época é composta por um ou mais lotes (BROWNLEE, 2018). Para este trabalho foram obtidos resultados significativos utilizando um número de épocas igual a 5000.

Após as iterações definidas pelo número de épocas, é utilizada a função *Softmax*, passando os valores de pesos que foram atualizados pelo treinamento para definir a probabilidade de a RNA realizar uma previsão correta.

Por fim, é comparado as previsões realizadas pela fatia de dados de treinamento com as previsões corretas, oriundas da fatia de testes da base de dados. Para mostrar a efetividade do treinamento e das previsões é impresso pelo algoritmo a taxa de acerto das previsões, ou seja, a taxa de acerto do algoritmo de reconhecer a classe de cada forma de onda.

Para a execução de todas as etapas acima citadas, o algoritmo levou um tempo de 2 minutos para rodar todo o treinamento e classificação dentro da arquitetura proposta com a configuração da máquina citada no início deste capítulo.

Parte III

Resultados e Conclusões

7 RESULTADOS

Este capítulo apresenta os resultados obtidos pelas execuções de diversos testes na RNA criada, onde foram testados parâmetros variados e analisado qual combinação apresentou melhores resultados. O resultado é medido considerando a taxa de acerto da RNA para a classificação da forma de onda, onde uma taxa com o valor "1" representa 100% de acerto na classificação após o treinamento.

7.1 Evolução da base de dados e relação com resultados

Os testes foram realizados com base em 4 fases, como dito na Seção 5.4, ao final dos testes obtivemos 428 vetores para representar as 8 classes de testes. Antes deste número final as fases de testes contiveram os seguintes números:

- **Primeira fase**, contou com 92 vetores os quais representavam 7 classes (nesta etapa ainda não era considerada a classe de tensão contínua);
- **Segunda fase**, partindo dos vetores contidos na fase anterior, foi acrescentada uma nova classe (tensão contínua), contendo 15 vetores para representar essa oitava classe, somando 107 vetores para representar as 8 classes;
- **Terceira fase**, os vetores contidos até a segunda fase foram dobrados, totalizando nesta fase 214 vetores para representar as 8 classes;
- **Quarta fase**, por fim para cada vetor criado até o momento foi inserido um ruído branco conforme mencionado na Seção 5.3.5, totalizando os 428 vetores finais que foram utilizados para representar as 8 classes de testes.

7.2 Resultados obtidos a cada fase de testes

Pelo fato de os pesos utilizados na rede neural serem iniciados de forma aleatória, para cada combinação de parâmetros foram realizadas 10 execuções de testes, definindo o resultado obtido como a média aritmética das 10 execuções. Os parâmetros alterados nas execuções dos testes foram o número de neurônios da camada oculta, número de épocas de treinamento, tamanho do lote e taxa de aprendizagem.

7.2.1 Resultados da primeira fase de testes

Na primeira fase de testes, foram realizadas 27 combinações de parâmetros para observar o comportamento da rede e resultados com base na alteração desses parâmetros, visto que foram realizadas 10 execuções, para obter a média dos resultados para cada combinação, um total de 270 execuções foram realizadas para validar a consistência dos resultados. O melhor e o pior resultado obtido dentre as execuções pode ser visto na Tabela 10.

Tabela 10 - Resultados na primeira fase de testes.

Número de Neurônios na camada Oculta	Taxa de Aprendizado	Tamanho do Lote	Número de Épocas	Média (taxa de acerto)
300	0,001	8	10000	0,94
170	0,001	6	10000	0,87

Fonte: Autor.

O desvio padrão calculado para a primeira fase de testes, dentro das 270 execuções foi de 0,07. As diversas configurações dos parâmetros da RNA não apresentam mudanças bruscas nos resultados obtidos. Desta forma uma arquitetura entregando uma taxa de acerto de 94%, como visto na Tabela 10 é um resultado satisfatório.

7.2.2 Resultados da segunda fase de testes

Para a segunda fase de testes, foi observado um resultado inferior a primeira fase, mesmo tendo mais vetores conhecidos na base de dados e também aumentando o número de combinações para 33. Nesta fase também foram feitas 10 execuções de testes, totalizando 330 execuções para obter os resultados contidos na Tabela 11, o qual apresenta o melhor e pior resultado obtido.

Tabela 11 - Resultados na segunda fase de testes.

Número de Neurônios na camada Oculta	Taxa de Aprendizado	Tamanho do Lote	Número de Épocas	Média (taxa de acerto)
165	0,001	5	10000	0,90
120	0,001	6	10000	0,78

Fonte: Autor.

O desvio padrão calculado para a segunda fase de testes, dentro das 330 execuções foi de 0,06. De maneira semelhante a primeira fase, é observado que as diversas configurações dos parâmetros da RNA não apresentam mudanças bruscas nos resultados obtidos. Nesta fase a arquitetura que apresentou o melhor resultado contou com uma taxa de acerto de 90%, sendo observado que a mudança na base de dados foi a principal causa para a queda neste percentual entre a primeira e segunda fase.

7.2.3 Resultados da terceira fase de testes

Pelo fato do resultado da segunda fase de testes ter sido inferior a primeira, o número de vetores foi dobrado, como mencionado na Seção 7.1, além disto para esta fase o número de combinações de parâmetros foi aumentado para 39, totalizando 390 execuções de testes. A Tabela 12 apresenta o melhor e pior resultado obtido na terceira fase de testes.

Tabela 12 - Resultados na terceira fase de testes.

Número de Neurônios na camada Oculta	Taxa de Aprendizado	Tamanho do Lote	Número de Épocas	Média (taxa de acerto)
140	0,001	12	5000	0,97
99	0,001	12	5000	0,92

Fonte: Autor.

O desvio padrão calculado para a terceira fase de testes, dentro das 390 execuções foi de 0,03. O aumento da taxa de acerto para 97% como visto na Tabela 12, pode ser observado pelo aumento dos dados conhecidos para treinamento disponibilizados para a RNA, com isso é concluído que a alteração dos parâmetros da RNA tem pouca influência na taxa de acerto final e que a fatia de dados de treinamento é um passo importante para obtenção de melhores resultados.

7.2.4 Resultados da quarta fase de testes

Na última fase de testes foram realizadas 39 combinações totalizando também, 390 execuções de testes, formando a base de dados considera final, contendo nesta fase 428 vetores para a representação das 8 classes. O melhor e pior resultado das execuções são apresentados na Tabela 13.

Tabela 13 - Resultados na quarta fase de testes.

Número de Neurônios na camada Oculta	Taxa de Aprendizado	Tamanho do Lote	Número de Épocas	Média (taxa de acerto)
99	0,0001	10	5000	0,92
120	0,0001	20	5000	0,87

Fonte: Autor.

Na última fase o desvio padrão calculado, dentro das 390 execuções foi de 0,02 o que mostra que o aumento da base de dados torna o desvio padrão menor. O decréscimo da taxa de acerto como visto na Tabela 13 era esperado por conta da inserção de ruído, porém uma taxa de acerto de 92% pode ser considerada um bom resultado, mostrando que mesmo tendo uma base com dados ruidosos a RNA tem grande capacidade de classificar os tipos de ruídos apresentados a ela.

8 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho teve como principais objetivos: criar uma rede neural artificial capaz de identificar e classificar tipos de ruídos conduzidos presentes nas linhas de alimentação DC em dispositivos eletrônicos, possibilitando um cenário futuro de testes inteligentes integrando execuções autônomas de testes para diversos cenários.

Para alcançar os propósitos deste trabalho, as seguintes etapas foram implementadas:

1. Pesquisar e estudar as normas IEC 61000-4-2, IEC 61000-4-4, IEC 61000-4-5 e IEC 61000-4-29 e documentos técnicos consagrados da literatura que descrevem os testes de EMC e EMI nos dispositivos eletrônicos;
2. Criar uma base de dados (*Dataset*) contento as formas de ondas descritas pelas normas relacionadas a este trabalho. Contribuindo para a comunidade de desenvolvedores;
3. Modelar e implementar uma RNA do tipo MLP *feedforward* com aprendizagem por *Backpropagation* para classificar as formas de ondas das normas.

Realizar testes de estresse alterando os parâmetros da RNA, a fim de extrair a melhor configuração do modelo. Ao todo, 1380 execuções da RNA foram realizadas para chegar aos resultados finais.

Portanto, com base nas análises e resultados obtidos, é concluído que a evolução das fases de testes mostrou que a medida que o número de conjuntos de vetores fornecidos para a RNA aumenta, a taxa de acerto aumenta proporcionalmente. Já a alteração dos parâmetros da RNA tem um peso menor no aumento da taxa de acerto nas previsões e classificações. Sendo assim, quanto maior for o número de amostras fornecidas para o treinamento da RNA, a probabilidade de acerto na classificação do tipo do ruído tende a aumentar, ou seja, uma base de dados sólida é fundamental.

Este trabalho também validou a hipótese da inserção da inteligência artificial e aprendizado de máquina junto a testes de dispositivos eletrônicos, abrindo portas para que a metodologia aplicada neste trabalho possa contribuir para novos cenários de testes, apoiando o desenvolvimento de dispositivos eletrônicos robustos e confiáveis.

8.1 Propostas para trabalhos futuros

Visto que este trabalho validou a hipótese de uma rede neural reconhecer as formas de ondas propostas pelas normas IEC 61000-4-2, IEC 61000-4-4, IEC 61000-4-5 e IEC 61000-4-29 e criou uma base de dados válida para utilização em RNAs, sugere-se para trabalhos futuros a abrangência de outras normas, aumentando a base de dados para treinamentos em RNAs, a fim de aumentar as contribuições para a comunidade de desenvolvedores.

Uma outra alternativa de trabalho futuro, seria a implementação do modelo desenvolvido neste trabalho de forma embarcada em um microprocessor, criando um cenário automatizado de testes e abrangendo possibilidades de “auto-testes” com execuções e criações de relatórios de execução e comportamento dado determinado ensaio.

Como solução alternativa à implementação da RNA para a geração de código a ser executado por um microprocessor, sugere-se também a possibilidade de implementação da RNA em linguagem VHDL para mapeamento direto em hardware dedicado de dispositivos lógico-programáveis (FPGAs). Em ambos os casos, o processador ou o FPGA seriam componentes instanciados ao nível de placa, possibilitando o aumento da resiliência/tolerância a falhas do sistema embarcado alvo ao ruído eletromagnético, em tempo real. Espera-se que neste cenário, o sistema seria capaz de realizar ações pró-ativas, tais como reduzir a frequência de relógio do CI, aumentar a tensão da fonte de alimentação, utilizar técnicas de redundância no tempo e/ou ativar códigos de detecção e correção de erros (EDAC) durante o período em que o sistema estiver operando sob tal exposição ao ruído.

Suponha, por exemplo, que para tipos de ruído que afetam um sistema crítico de tempo real por curtos períodos (por exemplo, quedas de tensão, interrupções curtas e variações de tensão: IEC 61000-4-29 std), talvez seja preferível executar ações de *redundância de tempo* em vez de *reduzir a frequência do relógio* (uma vez que o tempo é o parâmetro crítico). Ao contrário, se este sistema for exposto a ruído do tipo *burst* de longo período (IEC 61000-4-4), seria preferível *diminuir a frequência do relógio* do sistema em vez de executar *redundância de tempo*. Nesta última abordagem, o processador executa três vezes o mesmo trecho de código, nos instantes de tempo t_1 , t_2 e t_3 , e realiza a votação majoritária para produzir uma saída livre de falhas. Observe que, neste caso, a redundância de tempo é uma opção inválida, pois há uma grande probabilidade de que o processador execute o mesmo trecho de código nos instantes de tempo t_1 , t_2 e t_3 enquanto estiver exposto ao ruído (do tipo *rajada de longo período*) e assim, propenso a produzir cálculos errôneos em qualquer um dos três instantes t_1 , t_2 e t_3 .

Essas ações devem ser executadas apenas durante o período em que o sistema está sofrendo com o ruído de entrada. Uma vez que tal ocorrência de ruído desapareça, as ações ativadas são automaticamente desfeitas e o sistema retorna à operação normal.

Resultados experimentais sugerem que a abordagem proposta pode ser muito eficaz na identificação de diferentes tipos de ruído conduzidos nas linhas de alimentação de um sistema durante a operação em campo.

Além disso, um sistema projetado tendo em mente a abordagem proposta seria um forte candidato a ser aprovado pela regulamentação oficial do governo, onde as normas IEC fornecem a principal orientação para a certificação.

REFERÊNCIAS

- ALBERTO, Carlos; SILVA, Albuquerque. **IMPLEMENTAÇÃO DE UMA MATRIZ DE NEURÔNIOS DINAMICAMENTE E PARCIALMENTE RECONFIGURÁVEL PARA DESCRIÇÃO DE TOPOLOGIAS DE REDES NEURAIS ARTIFICIAIS MULTILAYER PERCEPTRONS**. 83 f. 2015. Tese (Doutorado em Engenharia Elétrica) - Universidade Federal do Rio Grande do Norte, Natal, 2015. Disponível em: <https://repositorio.ufrn.br/handle/123456789/21138>
- AMRUTHA, J.; REMYA AJAI, A. S. Performance analysis of backpropagation algorithm of artificial neural networks in verilog. **2018 3rd IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology, RTEICT 2018 - Proceedings**, [s. l.], p. 1547–1550, 2018. Disponível em: <https://doi.org/10.1109/RTEICT42901.2018.9012614>
- ANACONDA. **Anaconda Navigator**. [S. l.], 2021. Disponível em: <https://docs.anaconda.com/anaconda/navigator/index.html>. Acesso em: 5 nov. 2021.
- AVIŽIENIS, Algirdas *et al.* Basic concepts and taxonomy of dependable and secure computing. **IEEE Transactions on Dependable and Secure Computing**, [s. l.], v. 1, n. 1, p. 11–33, 2004. Disponível em: <https://doi.org/10.1109/TDSC.2004.2>
- BALESTRASSI, Pedro Paulo. **IDENTIFICAÇÃO DE PADRÕES EM GRÁFICOS DE CONTROLE ESTATÍSTICO DE PROCESSOS, EM TEMPO REAL, UTILIZANDO SÉRIES TEMPORAIS E REDES NEURAIS ARTIFICIAIS**. 2000. Tese (Doutorado em Engenharia de Produção) - Universidade Federal de Santa Catarina, Florianópolis, 2000. Disponível em: <http://repositorio.ufsc.br/xmlui/handle/123456789/78244>
- BAUER, Susanne; DEUTSCHMANN, Bernd; WINKLER, Gunter. Prediction of the robustness of integrated circuits against EFT/BURST. **IEEE International Symposium on Electromagnetic Compatibility**, [s. l.], v. 2015-Septm, p. 45–49, 2015. Disponível em: <https://doi.org/10.1109/ISEMC.2015.7256130>
- BEN DHIA, Sonia, RAMDANI, Mohamed, SICARD, Etienne. **Electromagnetic Compatibility of Integrated Circuits: Techniques for low emission and susceptibility**. 1. ed. Springer Publishing Company, 2006. 473 p. v. 1. ISBN 978-1-4614-9831-5.
- BENFICA, Juliano D’Ornelas. **Plataforma para Desenvolvimento de SoC (System-on-Chip) Robusto a Interferência Eletromagnética**. 2007. Dissertação (Mestrado) - Pontifícia Universidade Católica do Rio Grande do Sul - PUCRS, Faculdade de Engenharia, Programa de Pós-Graduação em Engenharia Elétrica, Porto alegre, 2007.
- BENFICA, Juliano D’Ornelas. **Plataforma para testes e qualificação de dispositivos reconfiguráveis e sistemas em chip, submetidos aos efeitos combinados da interferência eletromagnética e da radiação ionizante**. 2015. Tese (Doutorado em Engenharia Elétrica) - Universidade Federal de Santa Catarina, Santa Catarina, 2015.
- BENFICA, 2020:** Juliano Benfica, Fabian Vargas, Matheus Fay Soares, Dorian Schramm, “Conducted EMI Susceptibility Analysis of a COTS Processor as Function of Aging”, *Microelectronics Reliability Journal*, Vol. October 2020, Elsevier, ISSN: 0026-2714, **DOI:**

<https://doi.org/10.1016/j.microrel.2020.113884>.

BORSOI, Sad Sandrini. **Coordenação de dispositivos de proteção contra surtos em baixa tensão: ênfase instalações nucleares**. 2018. - Universidade de São Paulo, São Paulo, 2018. Disponível em: <http://www.teses.usp.br/teses/disponiveis/3/3143/tde-27022018-082038/en.php>

BRAGA, Antonio de Padua; CARVALHO, Andre Ponce de Leon F. de; LUDERMIR, Teresa Bernarda. **Redes Neurais Artificiais - Teoria e Aplicações**. 2nd. ed. [S. l.]: LTC, 2007.
BROWNLEE, Jason. **Difference Between a Batch and an Epoch in a Neural Network**. [S. l.], 2018. Disponível em: <https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/>. Acesso em: 27 dez. 2021.

BUZDUGAN, Mircea Ion; BALAN, Horia. A brief review of transient electromagnetic immunity testing. **CANDO-EPE 2019 - Proceedings: IEEE 2nd International Conference and Workshop in Obuda on Electrical and Power Engineering**, [s. l.], p. 127–132, 2019. Disponível em: <https://doi.org/10.1109/CANDO-EPE47959.2019.9111040>

CAROBBI, Carlo F.M. *et al.* Time-domain characterization of the surge, EFT/burst, and ESD measurement systems. **IEEE Transactions on Instrumentation and Measurement**, [s. l.], v. 62, n. 6, p. 1840–1846, 2013. Disponível em: <https://doi.org/10.1109/TIM.2013.2239017>

COLAÇO, D. F. *et al.* Sistema de Monitoramento de Chaves Seccionadoras por Visão Computacional e Redes Neurais Artificiais para Suporte ao Sistema de Controle Supervisório da CHESF. **IV Congresso de inovação tecnologica em energia eletrica**, [s. l.], n. January, p. 1–10, 2007. Disponível em: <https://doi.org/10.13140/2.1.2111.9689>

DANTAS, Jamilson Ramalho. **Modelos para Análise de Dependabilidade de Arquiteturas de Computação em Nuvem**. 85 f. 2013. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Pernambuco, Recife, 2013. Disponível em: <https://repositorio.ufpe.br/handle/123456789/12373>

DE MEDEIROS, André Rodrigues. **MOVIMENTO MAKER E INTELIGÊNCIA ARTIFICIAL: desenvolvimento de um protótipo reproduzível e aplicável**. 2021. Dissertação (Mestrado) - Universidade Estadual Paulista, Bauru, 2021. Disponível em: <http://hdl.handle.net/11449/204236>

ESCOVEDO, Tatiana. **Implementando um Modelo de Classificação no Scikit-Learn**. [S. l.], 2019. Disponível em: <https://tatianaesc.medium.com/implementando-um-modelo-de-classificacao-no-scikit-learn-6206d684b377?p=6206d684b377>. Acesso em: 1 dez. 2021.

ESTEVAN, 2021: Estevan L. Lara, Allan A. Constante, Juliano Benfica, Fabian Vargas, Alexandre Boyer, Sonia B. Dhia, Andreas Gleinser, Gunter Winkler, Bernd Deutschmann, "Impact of Place and Route Strategy on FPGA Electromagnetic Emission". *Microelectronics Reliability Journal*, Elsevier, Available online 11 October 2021, ISSN: 0026-2714, DOI: <https://doi.org/10.1016/j.microrel.2021.114333>.

FILHO, Paulo de Tarso Salviano. **Ferramenta didática para o ensino de vibrações e de dinâmica em engenharia paulo de tarso salviano filho**. 111 f. 2009. Dissertação (Mestrado) - Universidade de Brasília - Faculdade de Tecnologia, Brasília, 2009.

FOUNTAS, Zafeirios. Imperial College Spiking Neural Networks for Human-like Avatar Control in a Department of Computing Spiking Neural Networks for Human-like Avatar Control in a Simulated Environment by. [s. l.], n. October, 2015.

GOERL, 2019: Roger Goerl, Paulo Ricardo Cechelero Villa, Fabian Vargas, César Augusto Missio Marcon, Nilberto H. Medina, Nemitala Added, Marcilei Aparecida G. da Silva, “Combined ionizing radiation & electromagnetic interference test procedure to achieve reliable integrated circuits”, *Microelectronics Reliability Journal*, Elsevier, p.113341, 2019. Online publication: 23-Sept-2019, DOI: [10.1016/j.microrel.2019.06.033](https://doi.org/10.1016/j.microrel.2019.06.033).

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning (Adaptive Computation and Machine Learning series)**. [S. l.]: The MIT Press; Illustrated edition, 2016.

GRIFFITHS, Francis; OOI, Melanie. The fourth industrial revolution - Industry 4.0 and IoT [Trends in Future I and M]. **IEEE Instrumentation and Measurement Magazine**, [s. l.], v. 21, n. 6, p. 29–31, 2018. Disponível em: <https://doi.org/10.1109/MIM.2018.8573590>

GUIMARÃES, Edson da Silva. **Aprendizado de Máquina aplicado à predição da produtividade da cultura da soja utilizando dados de clima e solo**. 2019. - Universidade de São Paulo, São Paulo, 2019. Disponível em: <https://www.teses.usp.br/teses/disponiveis/55/55137/tde-09062020-123106/>

GUYON, I. Neural networks and applications tutorial. **Physics Reports**, [s. l.], v. 207, n. 3–5, p. 215–259, 1991. Disponível em: [https://doi.org/10.1016/0370-1573\(91\)90146-D](https://doi.org/10.1016/0370-1573(91)90146-D)
HAYKIN, Simon. **Neural Networks: A Comprehensive Foundation**. 2. ed. [S. l.]: Pearson, 1998.

HAYKIN, Simon. **Neural Networks and Learning Machines**. 3rd edition. [S. l.]: Pearson, 2008.

HAYKIN, Simon. **Redes Neurais - Princípios e Prática**. 2. ed. [S. l.]: Bookman, 2001.

HELEGDA, Lara Colognese. **Análise da associação entre interleucina-6 e doença cardiovascular e a busca de um modelo em redes neurais artificiais para identificação de risco cardiovascular em pacientes com síndrome metabólica**. 131 f. 2014. Tese (Doutorado em Medicina) - Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2014. Disponível em: <http://tede2.pucrs.br/tede2/handle/tede/1795>

HUANG, Hao. **Implementação de algoritmos de TensorFlow para detectar Patologias Cardíacas**. 2019. - Universidade do Minho, Braga, 2019. Disponível em: <http://hdl.handle.net/1822/65927>

IEC, International Electrotechnical Commission. **IEC 61000-4-2. Electromagnetic compatibility (EMC) – Part 4-2: Testing and measurement techniques – Electrostatic discharge immunity test**. [S. l.: s. n.], 2001.

IEC, International Electrotechnical Commission. **IEC 61000-4-29. Electromagnetic compatibility (EMC) – Part 4-29: Testing and measurement techniques – Voltage dips**,

short interruptions and voltage variations on d.c. input power port immunity tests. [S. l.: s. n.], 2000.

IEC, International Electrotechnical Commission. **IEC 61000-4-4. Electromagnetic compatibility (EMC) – Part 4-4: Testing and measurement techniques – Electrical fast transient/burst immunity test.** [S. l.: s. n.], 2004.

IEC, International Electrotechnical Commission. **IEC 61000-4-5. Electromagnetic compatibility (EMC) - Part 4-5: Testing and measurement techniques – Surge immunity test.** [S. l.: s. n.], 2005.

IEC, International Electrotechnical Commission. **International Electrotechnical Vocabulary - Part 161 (60050-161).** Geneva, Switzerland: IEC - International Electrotechnical Commission, 1990.

IEC, International Electrotechnical Commission. **International Standard IEC 61000-4-2: “Electromagnetic Compatibility (EMC), Part 4: Testing and Measurement Techniques, Section 2: Electrostatic Discharge Immunity Test.** [S. l.: s. n.], 2008.

IEC, International Electrotechnical Commission. **Who we are.** [S. l.], [s. d.]. Disponível em: <https://www.iec.ch/who-we-are?ref=menu>. Acesso em: 10 dez. 2021.

KATHAIL, Vinod *et al.* PICO: Automatically designing custom computers. **Computer**, [s. l.], v. 35, n. 9, p. 39–47, 2002. Disponível em: <https://doi.org/10.1109/MC.2002.1033026>

KATSIVELIS, Pavlos K. *et al.* Electrostatic discharge current linear approach and circuit design method. **Energies**, [s. l.], v. 3, n. 11, p. 1728–1740, 2010. Disponível em: <https://doi.org/10.3390/en3111728>

KINGMA, Diederik P.; BA, Jimmy Lei. Adam: A method for stochastic optimization. **3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings**, [s. l.], p. 1–15, 2015.

KRAUS, John Daniel; FLEISCH, Daniel A. **Electromagnetics with applications.** 5th edition. [S. l.]: William C Brown Pub, 1999.

LAPRIE, J. Dependable computing: concepts, challenges. **FTCS’95: Proceedings of the Twenty-Fifth international conference on Fault-tolerant computing**, [s. l.], v. 1, p. 42–54, 1995.

LAPRIE, Jean Claude. Dependability: From concepts to limits. **Proceedings of the IFIP International Workshop on Dependable Computing and its Applications. DCIA 98**, [s. l.], p. 108–126, 1998.

LAU, Mian Mian; LIM, King Hann. Review of adaptive activation function in deep neural network. **2018 IEEE EMBS Conference on Biomedical Engineering and Sciences, IECBES 2018 - Proceedings**, [s. l.], p. 686–690, 2019. Disponível em: <https://doi.org/10.1109/IECBES.2018.08626714>

LAURENE, Fausett. **Fundamentals of Neural Networks: Architectures, Algorithms, and Applications.** [S. l.]: Prentice-Hall, Inc., 1994.

LECUN, Yann *et al.* Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, [s. l.], v. 86, n. 11, p. 2278–2323, 1998. Disponível em: <https://doi.org/10.1109/5.726791>

LEMENKOVA, Polina. Processing Oceanographic Data By Python Libraries Numpy, Scipy and Pandas. **Aquatic Research**, [s. l.], v. 2, n. 2, p. 73–91, 2019. Disponível em: <https://doi.org/10.3153/ar19009>

LIZ, Muriel Bittencourt de. **Introdução à compatibilidade eletromagnética em conversores estáticos**. 124 f. 1999. Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal de Santa Catarina, Florianópolis, 1999. Disponível em: <http://repositorio.ufsc.br/xmlui/handle/123456789/81098>

LORENSI, Lucas; SANTOS, D O S. **Desenvolvimento de um framework integrado de redes neurais artificiais e lógica difusa**. 99 f. 2008. Dissertação (Mestrado em Engenharia elétrica) - Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2008. Disponível em: <http://tede2.pucrs.br/tede2/handle/tede/3000>

MAS, J. F.; FLORES, J. J. The application of artificial neural networks to the analysis of remotely sensed data. **International Journal of Remote Sensing**, [s. l.], v. 29, n. 3, p. 617–663, 2008. Disponível em: <https://doi.org/10.1080/01431160701352154>

MIGUEZ, Geraldo Azar. **OTIMIZAÇÃO DO ALGORITMO DE BACKPROPAGATION PELO USO DA FUNÇÃO DE ATIVAÇÃO BI-HIPERBÓLICA**. 110 f. 2012. Tese (Doutorado em Ciências da Computação) - Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2012.

MODESTI, Paulo Henrique. **MÉTODO BASEADO EM INTELIGÊNCIA ARTIFICIAL PARA PREVISÃO DO PRAZO DE ENTREGA DE TAREFAS EM ESTAÇÕES DE MANUFATURA**. 2020. - Universidade Tecnológica Federal do Parana, Curitiba, 2020. Disponível em: <http://repositorio.utfpr.edu.br/jspui/handle/1/25001>

MORAES, Marlon Leandro. **VALIDAÇÃO DE UMA TÉCNICA PARA O AUMENTO DA ROBUSTEZ DE SoC ' s A FLUTUAÇÕES DE VALIDAÇÃO DE UMA TÉCNICA PARA O AUMENTO DA ROBUSTEZ DE SoC ' s A FLUTUAÇÕES DE**. 2008. Dissertação (Mestrado em Engenharia Elétrica) - Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2008. Disponível em: <http://tede2.pucrs.br/tede2/handle/tede/2985>

MORGAN, David. **A Handbook for EMC Testing and Measurement (Materials, Circuits and Devices)**. [S. l.]: The Institution of Engineering and Technology, 1994.

NOURANI, Mehrdad; ATTARHA, Amir. Signal Integrity: Fault Modeling and Testing in High-Speed SoCs. **Journal of Electronic Testing**, [s. l.], 2002. Disponível em: <https://doi.org/10.1023/A:1016514129296>

NWANKPA, Chigozie *et al.* Activation Functions: Comparison of trends in Practice and Research for Deep Learning. [s. l.], p. 1–20, 2018. Disponível em: <http://arxiv.org/abs/1811.03378>

OTT, Henry W. **Electromagnetic compatibility engineering**. 1st editioed. [S. l.]: Wiley, 2009.

PANDAS. **Python Data Analysis Library**. [S. l.], 2018. Disponível em: <https://pandas.pydata.org/index.html>. Acesso em: 16 nov. 2021.

PAUL, Clayton R. **Introduction to Electromagnetic Compatibility**. 2nd edition. [S. l.]: Wiley-Interscience, 2006.

PÉREZ, Nicolás Louzán; DONSIÓN, Manuel Pérez. Technical methods for the prevention and correction of voltage sags and short interruptions inside the industrial plants and in the distribution networks. **Renewable Energy and Power Quality Journal**, [s. l.], v. 1, n. 1, p. 628–635, 2003. Disponível em: <https://doi.org/10.24084/repqj01.442>

PRADHAN, Dhiraj K. **Fault-Tolerant Computer System Design**. United States: Prentice-Hall, Inc., 1996.

PRESTES, Darcio Pinto. **Plataforma Para Injeção De Ruído Eletromagnético Conduzido Em Circuitos Integrados**. 98 f. 2010. Dissertação (Mestrado em Engenharia Elétrica) - Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2010. Disponível em: <http://tede2.pucrs.br/tede2/handle/tede/3047>

PRYTULA, Leandro. **Desenvolvimento e análise de compatibilidade eletromagnética de equipamentos utilizados nas atividades de elevação de pessoas**. 2020. Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal do Rio Grande do Sul, Porto Alegre, 2020. Disponível em: <http://hdl.handle.net/10183/215313>

PSCAD, Web Help. **Standard Surge Waveforms**. [S. l.], [s. d.]. Disponível em: [https://www.pscad.com/webhelp/ol-help.htm#Master_Library_Models/CSMF/Surge_Generators/Wavelet_Transformation_\(WT\).htm](https://www.pscad.com/webhelp/ol-help.htm#Master_Library_Models/CSMF/Surge_Generators/Wavelet_Transformation_(WT).htm). Acesso em: 10 dez. 2021.

RAIZER, Adroaldo. **Compatibilidade Eletromagnética**. Florianópolis: [s. n.], 2005. Disponível em: [https://paginas.fe.up.pt/~ee05161/ficheiros/artigos/Slides_CEM - Adroaldo Raizer.pdf](https://paginas.fe.up.pt/~ee05161/ficheiros/artigos/Slides_CEM_-_Adroaldo_Raizer.pdf)

RASAMOELINA, Andrinandrasana David; ADJAILIA, Fouzia; SINCAK, Peter. A Review of Activation Function for Artificial Neural Network. **SAMI 2020 - IEEE 18th World Symposium on Applied Machine Intelligence and Informatics, Proceedings**, [s. l.], p. 281–286, 2020. Disponível em: <https://doi.org/10.1109/SAMI48414.2020.9108717>

RASIT O. TOPALOGLU. **More than Moore Technologies for Next Generation Computer Design**. New York: Springer Publishing Company, Incorporated, 2015.

RODRIGUES, Dhileane De Andrade. **Mecanismo de Tolerância a Falhas para o EICIDS : Sistema de detecção de intrusão elástico e interno baseado em nuvem Mecanismo de Tolerância a Falhas para o EICIDS : Sistema de detecção de intrusão elástico e interno baseado em nuvem**. 2014. Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal do Maranhão, Sao Luís, 2014. Disponível em: <http://tede2.ufma.br:8080/jspui/handle/tede/521>

RUMELHART, David E.; HINTON, Geoffrey E.; WILLIAMS, Ronald J. Learning representations by back-propagating errors. **Nature**, [s. l.], v. 323, n. 6088, p. 533–536, 1986. Disponível em: <https://doi.org/10.1038/323533a0>

SANTANA, Alessandro Alves. **Programas em MATLAB para Implementação de Exemplos em Discretização de Equações Diferenciais Parciais**. 181 f. 1998. Dissertação (Mestrado) - Universidade de São Paulo, São Carlos, 1998.

SANTOS, André Pires. **Análise e comparação do desempenho de diferentes arquiteturas de redes neurais artificiais profundas aplicadas à triagem de lesões de pele**. 106 f. 2019. Dissertação (Mestrado em Engenharia Elétrica) - Universidade Presbiteriana Mackenzie, São Paulo, 2019. Disponível em: <http://tede.mackenzie.br/jspui/handle/tede/4232>

SANTOS, Francisco Lledo dos. **Redes neurais artificiais artmap-fuzzy aplicadas ao estudo de agitação marítima e ondas de lagos**. 257 f. 2013. Tese (Doutorado em Engenharia Elétrica) - Universidade Estadual Paulista, São Paulo, 2013. Disponível em: <https://repositorio.unesp.br/handle/11449/110523>

SANTOS, Gustavo Carvalho. **Algoritmos De Machine Learning Para Previsão De Ações Da B3**. 94 f. 2020. - Universidade Federal de Uberlândia, Uberlândia, 2020. Disponível em: <https://repositorio.ufu.br/handle/123456789/29897>

SILVA, Leandro Nunes de Castro. **ANÁLISE E SÍNTESE DE ESTRATÉGIAS DE APRENDIZADO PARA REDES NEURAIIS ARTIFICIAIS**. 1998. Dissertação (Mestrado em Engenharia Elétrica) - Universidade Estadual de Campinas, Campinas, 1998. Disponível em: <http://repositorio.unicamp.br/jspui/handle/REPOSIP/259070>

SIQUEIRA, Lindonete; ABDELOUAHAB, Zair. A fault tolerance mechanism for network intrusion detection system based on intelligent agents (NIDIA). **Proc. - The Fourth IEEE Workshop on Software Technol. for Future Embedded and Ubiquitous Syst., SEUS 2006 and the Second Int. Workshop on Collaborative Comput., Integr., and Assur., WCCIA 2006**, [s. l.], v. 2006, p. 49–54, 2006. Disponível em: <https://doi.org/10.1109/SEUS-WCCIA.2006.4>

SOARES, Matheus Fay. **Estudo dos efeitos combinados da interferência eletromagnética e envelhecimento na confiabilidade do microcontrolador Cortex-M4**. 2021. Dissertação (Mestrado em Engenharia Elétrica) - Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2021. Disponível em: <https://hdl.handle.net/10923/17325>

SOARES, 2021a: Matheus Fay Soares, Fabian Luis Vargas, Juliano D’Ornelas Benfica, "Conducted EMI Susceptibility Analysis of a COTS Processor as Function of Thermal Cycling and Overvoltage Stresses". *Microelectronics Reliability Journal*, Elsevier, Available online 11 October 2021, ISSN: 0026-2714, DOI: <https://doi.org/10.1016/j.microrel.2021.114247>.

TURITSYNA, E G; WEBB, S. Simple design of FBG-based VSB filters for ultra-dense WDM transmission *ELECTRONICS LETTERS* 20th January 2005. **Electronics letters**, [s. l.], v. 41, n. 2, p. 40–41, 2005. Disponível em: <https://doi.org/10.1049/el>

WANG, Kai *et al.* Numerical modeling of electrostatic discharge generators. **IEEE Transactions on Electromagnetic Compatibility**, [s. l.], v. 45, n. 2, p. 258–271, 2003. Disponível em: <https://doi.org/10.1109/TEMC.2003.810817>

WEBER, Taisy Silva. **Um roteiro para exploração dos conceitos básicos de tolerância a**

falhas. [S. l.]: Programa de PósGraduação em Computação – Instituto de Informática – UFRGS, 2003.

XU, Min; DAVID, Jeanne M.; KIM, Suk Hi. The fourth industrial revolution: Opportunities and challenges. **International Journal of Financial Research**, [s. l.], v. 9, n. 2, p. 90–95, 2018. Disponível em: <https://doi.org/10.5430/ijfr.v9n2p90>

APÊNDICE A – Código principal MATLAB

```

clc;
clear;

amostras = 191;
% Número de parametros na tabela
nQuedas = 36;
nShorts = 24;
nVariations = 36;
nTransients = 16;
nSourceOpen = 24;
nSourceShort = 24;
nDischarge = 24;
nConstante = 30;

voltageDips = cell(1,nQuedas);
voltageDipsRuido = cell(1,nQuedas);
shortInterruption = cell(1,nShorts);
shortInterruptionRuido = cell(1,nShorts);
voltageVariation = cell(1,nVariations);
voltageVariationRuido = cell(1,nVariations);
transientes = cell(1,nTransients);
transientesRuido = cell(1,nTransients);
sourceOpen = cell(1,nSourceOpen);
sourceOpenRuido = cell(1,nSourceOpen);
sourceShort = cell(1,nSourceShort);
sourceShortRuido = cell(1,nSourceShort);
discharge = cell(1,nDischarge);
dischargeRuido = cell(1,nDischarge);
constante = cell(1,nConstante);
constanteRuido = cell(1,nConstante);

% IEC 61000-4-29 Gerar voltage dips
voltageDips{1} = gerarDips(3.3,40,0.1,2);
voltageDips{2} = gerarDips(3.3,40,0.3,2);
voltageDips{3} = gerarDips(3.3,40,1,2);
voltageDips{4} = gerarDips(3.3,70,0.1,2);
voltageDips{5} = gerarDips(3.3,70,0.3,2);
voltageDips{6} = gerarDips(3.3,70,1,2);
voltageDips{7} = gerarDips(3.3,40,0.1,2);
voltageDips{8} = gerarDips(3.3,40,0.3,2);
voltageDips{9} = gerarDips(3.3,40,1,2);
voltageDips{10} = gerarDips(3.3,70,0.1,2);
voltageDips{11} = gerarDips(3.3,70,0.3,2);
voltageDips{12} = gerarDips(3.3,70,1,2);
voltageDips{13} = gerarDips(3.33,40,0.1,2);
voltageDips{14} = gerarDips(3.33,40,0.3,2);
voltageDips{15} = gerarDips(3.33,40,1,2);
voltageDips{16} = gerarDips(3.33,70,0.1,2);
voltageDips{17} = gerarDips(3.33,70,0.3,2);
voltageDips{18} = gerarDips(3.33,70,1,2);
voltageDips{19} = gerarDips(3.3,40,0.1,2);
voltageDips{20} = gerarDips(3.3,40,0.3,2);
voltageDips{21} = gerarDips(3.3,40,1,2);
voltageDips{22} = gerarDips(3.3,70,0.1,2);
voltageDips{23} = gerarDips(3.3,70,0.3,2);
voltageDips{24} = gerarDips(3.3,70,1,2);

```

```

voltageDips{25} = gerarDips (3.3,40,0.1,2);
voltageDips{26} = gerarDips (3.3,40,0.3,2);
voltageDips{27} = gerarDips (3.3,40,1,2);
voltageDips{28} = gerarDips (3.3,70,0.1,2);
voltageDips{29} = gerarDips (3.3,70,0.3,2);
voltageDips{30} = gerarDips (3.3,70,1,2);
voltageDips{31} = gerarDips (3.33,40,0.1,2);
voltageDips{32} = gerarDips (3.33,40,0.3,2);
voltageDips{33} = gerarDips (3.33,40,1,2);
voltageDips{34} = gerarDips (3.33,70,0.1,2);
voltageDips{35} = gerarDips (3.33,70,0.3,2);
voltageDips{36} = gerarDips (3.33,70,1,2);

%% Gerar Short interruption
shortInterruption{1} = gerarDips (3.3,100,0.1,2);
shortInterruption{2} = gerarDips (3.3,100,0.3,2);
shortInterruption{3} = gerarDips (3.3,100,1,2);
shortInterruption{4} = gerarDips (3.3,100,0.1,2);
shortInterruption{5} = gerarDips (3.3,100,0.3,2);
shortInterruption{6} = gerarDips (3.3,100,1,2);
shortInterruption{7} = gerarDips (3.33,100,0.1,2);
shortInterruption{8} = gerarDips (3.33,100,0.3,2);
shortInterruption{9} = gerarDips (3.33,100,1,2);
shortInterruption{10} = gerarDips (3.33,100,0.1,2);
shortInterruption{11} = gerarDips (3.33,100,0.3,2);
shortInterruption{12} = gerarDips (3.33,100,1,2);
shortInterruption{13} = gerarDips (3.3,100,0.1,2);
shortInterruption{14} = gerarDips (3.3,100,0.3,2);
shortInterruption{15} = gerarDips (3.3,100,1,2);
shortInterruption{16} = gerarDips (3.3,100,0.1,2);
shortInterruption{17} = gerarDips (3.3,100,0.3,2);
shortInterruption{18} = gerarDips (3.3,100,1,2);
shortInterruption{19} = gerarDips (3.33,100,0.1,2);
shortInterruption{20} = gerarDips (3.33,100,0.3,2);
shortInterruption{21} = gerarDips (3.33,100,1,2);
shortInterruption{22} = gerarDips (3.33,100,0.1,2);
shortInterruption{23} = gerarDips (3.33,100,0.3,2);
shortInterruption{24} = gerarDips (3.33,100,1,2);

% %% Gerar Voltage Variation
voltageVariation{1} = gerarDips (3.3,80,0.1,2);
voltageVariation{2} = gerarDips (3.3,80,0.3,2);
voltageVariation{3} = gerarDips (3.3,80,1,2);
voltageVariation{4} = gerarDips (3.3,120,0.1,2);
voltageVariation{5} = gerarDips (3.3,120,0.3,2);
voltageVariation{6} = gerarDips (3.3,120,1,2);
voltageVariation{7} = gerarDips (3.3,80,0.1,2);
voltageVariation{8} = gerarDips (3.3,80,0.3,2);
voltageVariation{9} = gerarDips (3.3,80,1,2);
voltageVariation{10} = gerarDips (3.3,120,0.1,2);
voltageVariation{11} = gerarDips (3.3,120,0.3,2);
voltageVariation{12} = gerarDips (3.3,120,1,2);
voltageVariation{13} = gerarDips (3.33,80,0.1,2);
voltageVariation{14} = gerarDips (3.33,80,0.3,2);
voltageVariation{15} = gerarDips (3.33,80,1,2);
voltageVariation{16} = gerarDips (3.33,120,0.1,2);
voltageVariation{17} = gerarDips (3.33,120,0.3,2);
voltageVariation{18} = gerarDips (3.33,120,1,2);
voltageVariation{19} = gerarDips (3.3,80,0.1,2);
voltageVariation{20} = gerarDips (3.3,80,0.3,2);

```

```

voltageVariation{21} = gerarDips (3.3,80,1,2);
voltageVariation{22} = gerarDips (3.3,120,0.1,2);
voltageVariation{23} = gerarDips (3.3,120,0.3,2);
voltageVariation{24} = gerarDips (3.3,120,1,2);
voltageVariation{25} = gerarDips (3.3,80,0.1,2);
voltageVariation{26} = gerarDips (3.3,80,0.3,2);
voltageVariation{27} = gerarDips (3.3,80,1,2);
voltageVariation{28} = gerarDips (3.3,120,0.1,2);
voltageVariation{29} = gerarDips (3.3,120,0.3,2);
voltageVariation{30} = gerarDips (3.3,120,1,2);
voltageVariation{31} = gerarDips (3.33,80,0.1,2);
voltageVariation{32} = gerarDips (3.33,80,0.3,2);
voltageVariation{33} = gerarDips (3.33,80,1,2);
voltageVariation{34} = gerarDips (3.33,120,0.1,2);
voltageVariation{35} = gerarDips (3.33,120,0.3,2);
voltageVariation{36} = gerarDips (3.33,120,1,2);

```

```

%% Gerar Transientes IEC 61000-4-4 fast transients/bursts.

```

```

transientes{1} = gerarTransient (250);
transientes{2} = gerarTransient (500);
transientes{3} = gerarTransient (1e3);
transientes{4} = gerarTransient (2e3);
transientes{5} = gerarTransient (250);
transientes{6} = gerarTransient (500);
transientes{7} = gerarTransient (1e3);
transientes{8} = gerarTransient (2e3);
transientes{9} = gerarTransient (250);
transientes{10} = gerarTransient (500);
transientes{11} = gerarTransient (1e3);
transientes{12} = gerarTransient (2e3);
transientes{13} = gerarTransient (250);
transientes{14} = gerarTransient (500);
transientes{15} = gerarTransient (1e3);
transientes{16} = gerarTransient (2e3);

```

```

%% Gerar Transientes IEC 61000-4-5 Source.

```

```

sourceOpen{1} = gerarSource (1,500);
sourceOpen{2} = gerarSource (1,1000);
sourceOpen{3} = gerarSource (1,2000);
sourceOpen{4} = gerarSource (1,4000);
sourceOpen{5} = gerarSource (1,500);
sourceOpen{6} = gerarSource (1,1000);
sourceOpen{7} = gerarSource (1,2000);
sourceOpen{8} = gerarSource (1,4000);
sourceOpen{9} = gerarSource (1,500);
sourceOpen{10} = gerarSource (1,1000);
sourceOpen{11} = gerarSource (1,2000);
sourceOpen{12} = gerarSource (1,4000);
sourceOpen{13} = gerarSource (1,500);
sourceOpen{14} = gerarSource (1,1000);
sourceOpen{15} = gerarSource (1,2000);
sourceOpen{16} = gerarSource (1,4000);
sourceOpen{17} = gerarSource (1,500);
sourceOpen{18} = gerarSource (1,1000);
sourceOpen{19} = gerarSource (1,2000);
sourceOpen{20} = gerarSource (1,4000);
sourceOpen{21} = gerarSource (1,500);
sourceOpen{22} = gerarSource (1,1000);
sourceOpen{23} = gerarSource (1,2000);

```

```
sourceOpen{24} = gerarSource (1,4000);
```

```
sourceShort{1} = gerarSource (2,250);
sourceShort{2} = gerarSource (2,500);
sourceShort{3} = gerarSource (2,1000);
sourceShort{4} = gerarSource (2,2000);
sourceShort{5} = gerarSource (2,250);
sourceShort{6} = gerarSource (2,500);
sourceShort{7} = gerarSource (2,1000);
sourceShort{8} = gerarSource (2,2000);
sourceShort{9} = gerarSource (2,250);
sourceShort{10} = gerarSource (2,500);
sourceShort{11} = gerarSource (2,1000);
sourceShort{12} = gerarSource (2,2000);
sourceShort{13} = gerarSource (2,250);
sourceShort{14} = gerarSource (2,500);
sourceShort{15} = gerarSource (2,1000);
sourceShort{16} = gerarSource (2,2000);
sourceShort{17} = gerarSource (2,250);
sourceShort{18} = gerarSource (2,500);
sourceShort{19} = gerarSource (2,1000);
sourceShort{20} = gerarSource (2,2000);
sourceShort{21} = gerarSource (2,250);
sourceShort{22} = gerarSource (2,500);
sourceShort{23} = gerarSource (2,1000);
sourceShort{24} = gerarSource (2,2000);
```

```
% Gerar Discharge IEC 61000-4-2.
```

```
discharge{1} = gerarDischarge (0.5);
discharge{2} = gerarDischarge (1);
discharge{3} = gerarDischarge (1.5);
discharge{4} = gerarDischarge (2);
discharge{5} = gerarDischarge (0.5);
discharge{6} = gerarDischarge (1);
discharge{7} = gerarDischarge (1.5);
discharge{8} = gerarDischarge (2);
discharge{9} = gerarDischarge (0.5);
discharge{10} = gerarDischarge (1);
discharge{11} = gerarDischarge (1.5);
discharge{12} = gerarDischarge (2);
discharge{13} = gerarDischarge (0.5);
discharge{14} = gerarDischarge (1);
discharge{15} = gerarDischarge (1.5);
discharge{16} = gerarDischarge (2);
discharge{17} = gerarDischarge (0.5);
discharge{18} = gerarDischarge (1);
discharge{19} = gerarDischarge (1.5);
discharge{20} = gerarDischarge (2);
discharge{21} = gerarDischarge (0.5);
discharge{22} = gerarDischarge (1);
discharge{23} = gerarDischarge (1.5);
discharge{24} = gerarDischarge (2);
```

```
% % Tensões constantes
```

```
constante{1} = gerarConstante (0,0);
constante{2} = gerarConstante (0,0);
constante{3} = gerarConstante (0,0);
constante{4} = gerarConstante (0,1);
```



```

constante{5} = gerarConstante(0,1);
constante{6} = gerarConstante(0,1);
constante{7} = gerarConstante(1,2);
constante{8} = gerarConstante(1,2);
constante{9} = gerarConstante(1,2);
constante{10} = gerarConstante(2,3.3);
constante{11} = gerarConstante(2,3.3);
constante{12} = gerarConstante(2,3.3);
constante{13} = gerarConstante(3.3,3.3);
constante{14} = gerarConstante(3.3,3.3);
constante{15} = gerarConstante(3.3,3.3);
constante{16} = gerarConstante(0,0);
constante{17} = gerarConstante(0,0);
constante{18} = gerarConstante(0,0);
constante{19} = gerarConstante(0,1);
constante{20} = gerarConstante(0,1);
constante{21} = gerarConstante(0,1);
constante{22} = gerarConstante(1,2);
constante{23} = gerarConstante(1,2);
constante{24} = gerarConstante(1,2);
constante{25} = gerarConstante(2,3.3);
constante{26} = gerarConstante(2,3.3);
constante{27} = gerarConstante(2,3.3);
constante{28} = gerarConstante(3.3,3.3);
constante{29} = gerarConstante(3.3,3.3);
constante{30} = gerarConstante(3.3,3.3);

%% Gerar o header

for i = 1:amostras
    if i == amostras
        names{i+1} = 'Forma de onda';
    end
    names{i} = ['Amostra ' num2str(i, '%d')];
end

header_string = names{1};
for i = 2:length(names)
    header_string = [header_string, ',', names{i}];
end

% write the string to a file

fid = fopen('dataset_iec.csv', 'w');
fprintf(fid, '%s\r\n', header_string);
fclose(fid);

for j = 1:nQuedas
    dlmwrite('dataset_iec.csv', voltageDips{j}, 'precision', '%.3f', '-
append')
end

for j = 1:nQuedas
    voltageDipsRuido{j} = awgn(voltageDips{j}, 30, 'measured');
    voltageDipsRuido{j}(length(voltageDipsRuido{j})) = 1;
    dlmwrite('dataset_iec.csv', voltageDipsRuido{j}, 'precision', '%.3f',
'-append')
end

```

```

for j = 1:nShorts
    dlmwrite('dataset_iec.csv', shortInterruption{j}, 'precision', '%.3f',
'-append')
end

for j = 1:nShorts
    shortInterruptionRuido{j} = awgn(shortInterruption{j},30,'measured');
    shortInterruptionRuido{j}(length(shortInterruptionRuido{j})) = 2;
    dlmwrite('dataset_iec.csv', shortInterruptionRuido{j}, 'precision',
'%.3f', '-append')
end

for j = 1:nVariations
    dlmwrite('dataset_iec.csv', voltageVariation{j}, 'precision', '%.3f',
'-append')
end

for j = 1:nVariations
    voltageVariationRuido{j} = awgn(voltageVariation{j},30,'measured');
    voltageVariationRuido{j}(length(voltageVariationRuido{j})) = 3;
    dlmwrite('dataset_iec.csv', voltageVariationRuido{j}, 'precision',
'%.3f', '-append')
end

for j = 1:nTransients
    dlmwrite('dataset_iec.csv', transientes{j}, 'precision', '%.3f', '-
append')
end

for j = 1:nTransients
    transientesRuido{j} = awgn(transientes{j},30,'measured');
    transientesRuido{j}(length(transientesRuido{j})) = 4;
    dlmwrite('dataset_iec.csv', transientesRuido{j}, 'precision', '%.3f',
'-append')
end

for j = 1:nSourceOpen
    dlmwrite('dataset_iec.csv', sourceOpen{j}, 'precision', '%.3f', '-
append')
end

for j = 1:nSourceOpen
    sourceOpenRuido{j} = awgn(sourceOpen{j},30,'measured');
    sourceOpenRuido{j}(length(sourceOpenRuido{j})) = 5;
    dlmwrite('dataset_iec.csv', sourceOpenRuido{j}, 'precision', '%.3f', '-
append')
end

for j = 1:nSourceShort
    dlmwrite('dataset_iec.csv', sourceShort{j}, 'precision', '%.3f', '-
append')
end

for j = 1:nSourceShort
    sourceShortRuido{j} = awgn(sourceShort{j},30,'measured');
    sourceShortRuido{j}(length(sourceShortRuido{j})) = 6;
    dlmwrite('dataset_iec.csv', sourceShortRuido{j}, 'precision', '%.3f',
'-append')
end

```

```
for j = 1:nDischarge
    dlmwrite('dataset_iec.csv', discharge{j}, 'precision', '%.3f', '-
append')
end

for j = 1:nDischarge
    dischargeRuido{j} = awgn(discharge{j},30,'measured');
    dischargeRuido{j}(length(dischargeRuido{j})) = 7;
    dlmwrite('dataset_iec.csv', dischargeRuido{j}, 'precision', '%.3f', '-
append')
end

for j = 1:nConstante
    dlmwrite('dataset_iec.csv', constante{j}, 'precision', '%.3f', '-
append')
end

for j = 1:nConstante
    constanteRuido{j} = awgn(constante{j},30,'measured');
    constanteRuido{j}(length(constanteRuido{j})) = 8;
    dlmwrite('dataset_iec.csv', constanteRuido{j}, 'precision', '%.3f', '-
append')
end
```

APÊNDICE B – Funções criadas para gerar formas de onda - MATLAB

```
function dips = gerarDips(ampl, queda, tempo, amostras)

t=0.1:0.01:amostras;

% adiciona a classe do sinal
if queda == 100
    classe = 2;
elseif queda <= 70
    classe = 1;
else
    classe = 3;
end

queda = ampl * (queda/100);
segundos = 0.5 + tempo;

if queda > 3.3
    dips=ampl*(heaviside(t))+(queda-3.3)*(heaviside(t-0.5))-(queda-
3.3)*(heaviside(t-segundos));
else
    dips=ampl*(heaviside(t))-queda*(heaviside(t-0.5))+queda*(heaviside(t-
segundos));
end

% adiciona a classe do sinal
if classe == 2
    dips(length(dips)+1) = 2;
elseif classe == 1
    dips(length(dips)+1) = 1;
else
    dips(length(dips)+1) = 3;
end

% figure()
% plot(t,dips)

end
```

```

function discharge = gerarDischarge(ganho)

%Make IEC 61000-4-2 current waveform

I1 = 16.6; %A
I2 = 9.3; %A
tau1 = 1.1e-9; %ns
tau2 = 2e-9; %ns
tau3 = 12e-9; %ns
tau4 = 37e-9; %ns
n = 1.8;
t=0:0.345e-9:65.6e-9;

i1 = (I1 / exp(-(tau1/tau2)*(n*tau2/tau1)^(1/n)) * ((t/tau1).^n .* ...
    exp(-t/tau2))./(1+(t/tau1).^n));

i2 = (I2 / exp(-(tau3/tau4)*(n*tau4/tau3)^(1/n)) * ((t/tau3).^n .* ...
    exp(-t/tau4))./(1+(t/tau3).^n));

discharge = ganho*(i1 + i2);

% adiciona a classe do sinal
discharge(length(discharge)+1) = 7;

% figure()
% plot(t,discharge)
% grid on
% grid minor
% set(gca,'XTick',[0 1e-8 2e-8 3e-8 4e-8 5e-8 6e-8 7e-8])
% set(gca,'XTickLabel',[0 10 20 30 40 50 60 70])
% xlabel('Tempo (ns)')
% ylabel('Tensão (kV)')

end

```

```
function source = gerarSource(type, ampl)

t = 0:0.4:76;
a1 = 0.0145;
b1 = 2.8353;
k1 = 1.0328;

tau = 3.911;
a3 = 0.01243;

if type == 1
    source = ampl*k1*(exp(-a1*t)-exp(-b1*t));
    % adiciona a classe do sinal
    source(length(source)+1) = 5;
else
    % source = -1*ampl*k2*(exp(-a2*t)-exp(-b2*t));% adiciona a classe do
sinal
    source = a3*ampl*(t.^3).*exp(-t/tau);
    % sourgeruido = awgn(source,30,'measured');
    source(length(source)+1) = 6;
end

% figure()
% plot(t,sourgeruido)
% grid on
% grid minor
% xlabel('Tempo (us)')
% ylabel('Tensão (V)')
% ylim([0 550])
end
```

```
function transient = gerarTransient(pico)

% n = 0:(1/200e3):0.95e-3;
n = 0.1:(1/100e3):0.10191;

transient = pico*heaviside(n);

for x = 1:(length(transient))
    if mod(x,2) ~= 1
        transient(x) = 0;
    end
    if x > 151
        transient(x) = 0;
    end
end

% adiciona a classe do sinal
transient(length(transient)) = 4;

% figure()
% plot(n,transient)
% grid on
% grid minor
% xlabel('Tempo (s)')
% ylabel('Tensão (kV)')
%
% ylim([0 0.26])

end
```

APÊNDICE C – Algoritmo para criação da RNA – Jupyter Notebook

```

import tensorflow as tf
import numpy as np
import pandas as pd

base = pd.read_csv('dataset_iec.csv')

x = base.drop('Forma de onda', axis=1) # atributos previsoers
y = base.iloc[:, -1] # pegando a ultima coluna para saber qual a classe

from sklearn.preprocessing import OneHotEncoder
onehot = OneHotEncoder(categories = 'auto')

y = y.values.reshape(-1,1)
y = onehot.fit_transform(y).toarray()

from sklearn.model_selection import train_test_split
x_treinamento, x_teste, y_treinamento, y_teste = train_test_split(x, y,
test_size = 0.5)

----((428, 191), (214, 191), (428, 8), (214, 8))----

neuronios_entrada = x.shape[1]
neuronios_oculta = 99
neuronios_saida = y.shape[1]

w = {'oculta': tf.Variable(tf.random_normal([neuronios_entrada,
neuronios_oculta])),
      'saida': tf.Variable(tf.random_normal([neuronios_oculta,
neuronios_saida]))}

b = {'oculta': tf.Variable(tf.random_normal([neuronios_oculta])),
      'saida': tf.Variable(tf.random_normal([neuronios_saida]))}

xph = tf.placeholder('float', [None, neuronios_entrada])
yph = tf.placeholder('float', [None, neuronios_saida])

def mlp(x, w, bias):
    camada_oculta = tf.add(tf.matmul(x, w['oculta']), b['oculta'])
    camada_oculta_ativacao = tf.nn.relu(camada_oculta)
    camada_saida = tf.add(tf.matmul(camada_oculta_ativacao, w['saida']),
b['saida'])
    return camada_saida

modelo = mlp(xph, w, b)

erro = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits =
modelo, labels = yph))
otimizador = tf.train.AdamOptimizer(learning_rate = 0.0001).minimize(erro)

batch_size = 10
batch_total = int(len(x_treinamento) / batch_size)

x_batches = np.array_split(x_treinamento, batch_total)

```



```

----
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for epoca in range(5001):
        erro_medio = 0.0
        batch_total = int(len(x_treinamento) / batch_size)
        x_batches = np.array_split(x_treinamento, batch_total)
        y_batches = np.array_split(y_treinamento, batch_total)
        for i in range(batch_total):
            x_batch, y_batch = x_batches[i], y_batches[i]
            _, custo = sess.run([otimizador, erro], feed_dict = {xph:
x_batch, yph: y_batch})
            erro_medio += custo / batch_total

        if epoca % 500 == 0:
            print('Época: ' + str(epoca+1) + ' erro: ' + str(erro_medio))

    w_final, b_final = sess.run([w,b])

    previsoes = tf.nn.softmax(mlp(xph, w_final, b_final))

    previsoes_corretas = tf.equal(tf.argmax(previsoes, 1), tf.argmax(yph,
1))

    taxa_acerto = tf.reduce_mean(tf.cast(previsoes_corretas, tf.float32))
print(sess.run(taxa_acerto, feed_dict = {xph: x_teste, yph: y_teste}))
----

```



Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Graduação
Av. Ipiranga, 6681 - Prédio 1 - 3º. andar
Porto Alegre - RS - Brasil
Fone: (51) 3320-3500 - Fax: (51) 3339-1564
E-mail: prograd@pucrs.br
Site: www.pucrs.br