UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
ENGENHARIA ELETRÔNICA E DE TELECOMUNICAÇÕES
CAMPUS PATOS DE MINAS

CAIO SÉJOUR ARAUJO

IDENTIFICAÇÃO DE ELETRODOMÉSTICOS COM BASE NO CONSUMO DE CORRENTE ELÉTRICA UTILIZANDO APRENDIZAGEM PROFUNDA

Patos de Minas - MG
2024

CAIO SÉJOUR ARAUJO

IDENTIFICAÇÃO DE ELETRODOMÉSTICOS COM BASE NO CONSUMO DE CORRENTE ELÉTRICA UTILIZANDO APRENDIZAGEM PROFUNDA

Trabalho de conclusão de curso apresentado à banca examinadora como requisito parcial de avaliação da disciplina de PFC2 da graduação em Engenharia Eletrônica e de Telecomunicações, da Faculdade de Engenharia Elétrica, da Universidade Federal de Uberlândia, Campus Patos de Minas.
Orientadora: Profa. Dra. Eliana Pantaleão

Patos de Minas - MG
2024

CAIO SÉJOUR ARAUJO

IDENTIFICAÇÃO DE ELETRODOMÉSTICOS COM BASE NO CONSUMO DE CORRENTE ELÉTRICA UTILIZANDO APRENDIZAGEM PROFUNDA

Trabalho de conclusão de curso apresentado à banca examinadora como requisito parcial de avaliação da disciplina de PFC2 da graduação em Engenharia Eletrônica e de Telecomunicações, da Faculdade de Engenharia Elétrica, da Universidade Federal de Uberlândia, Campus Patos de Minas.
Orientadora: Profa. Dra. Eliana Pantaleão

Patos de Minas, 04 de Julho de 2024

Banca Examinadora

Profa. Dra. Eliana Pantaleão - FACOM/UFU (Orientadora)

Prof. Dr. Pedro Luiz Lima Bertarini - FEELT/UFU (Membro 1)

Prof. Dr. Júlio Cézar Coelho - FEELT/UFU (Membro 2)

AGRADECIMENTOS

Ao meu pai Daniel Araujo, e à minha mãe Grade Adriana Szegeczky Séjour Araujo por todo apoio, incentivo, educação, carinho, ensinamentos e conselhos que ajudaram a formar quem eu sou e me fizeram poder chegar nesta etapa tão especial! Obrigado por tudo, amo vocês!

À minha companheira, Laryssa Aparecida Sales Barbosa, que foi como um pilar para mim durante este processo. Obrigado pelo carinho e apoio em todos os momentos, felizes e também difíceis, mas em que você sempre esteve presente, auxiliando e trazendo forças e motivação para que eu pudesse vencer todas as barreiras e conseguisse realizar esta grande conquista!

À Universidade Federal de Uberlândia *campus* Patos de Minas, em especial ao curso de Engenharia Eletrônica e de Telecomunicações, ao qual tive a grande satisfação de estudar, aprender e me desenvolver intelectualmente, e também como ser humano, durante toda a trajetória da minha graduação.

À minha orientadora Profa. Dra. Eliana Pantaleão, que me auxiliou neste trabalho, e também me proporcionou muitos ensinamentos durante o curso, além de apresentar e ensinar sobre a área em que eu mais me senti interessado e realizado durante a graduação, e que, provavelmente, seguirei trabalhando daqui por diante, que é a área de programação e desenvolvimento de software. Além disso, agradeço imensamente por apoiar desde o início a minha iniciativa empresarial, acreditando no projeto e ajudando para que ele se tornasse real!

A todos os meus professores da graduação, em especial aos membros da banca, Prof Dr. Júlio Cézar Coelho, e Prof Dr. Pedro Luiz Lima Bertarini, por todos os ensinamentos e pelo apoio nos projetos e nas aulas. Agradeço também por todas as oportunidades geradas por vocês, como o estágio, competições de robótica, eventos, congressos, entre vários outros momentos engrandecedores. Sou muito grato pela parceria e também pela amizade!

Aos amigos que fiz durante o curso, que fizeram com que essa jornada fosse divertida e prazerosa. Sou grato por todos os momentos de companheirismo, estudos, partilha, diversão e apoio. Com toda certeza, ao longo de toda a vida terei boas recordações dos diversos momentos que vivemos juntos.

RESUMO

O consumo de energia elétrica no mundo vem mudando a cada ano, de forma progressiva. As concessionárias de energia tem uma preocupação recorrente em suprir a demanda cada vez mais crescente, tentando evitar o racionamento, o uso de termelétricas, ou até mesmo os popularmente chamados "apagões". Tais acontecimentos se refletem também numa preocupação por parte do consumidor final, uma vez que a energia tem seu custo elevado nestes cenários, e, num pior dos casos, a interrupção dos serviços. Este trabalho tem como objetivo estudar uma implementação prática para trazer a informação sobre quais eletrodomésticos mais consomem energia dentro de um ambiente residencial, por meio de inteligência artificial. Outra proposta, é poder oferecer também a informação sobre o valor da conta de luz, antes mesmo de ser informada pela concessionária. Aliados, estes dados podem entregar, de forma objetiva, o controle de um consumo mais consciente por parte da população. Para que tal implementação seja realizada, foram utilizadas técnicas de *Machine Learning*, como Redes Neurais Artificiais, para analisar a corrente elétrica dos aparelhos residenciais. A viabilização desta implementação foi validada de acordo com a hipótese de uma característica no consumo de corrente dos aparelhos, em que, devido às suas especificidades de construção e circuitaria interna, possuem um padrão, peculiar de cada um, na forma de onda da corrente consumida, o que gera uma espécie de "assinatura" espectral, quando analisada no domínio da frequência. Foram realizados experimentos e coletas de dados por meio de equipamentos laboratoriais específicos ligados a eletrodomésticos comuns, com intuito de treinar uma rede neural, para que a mesma possa, com base na corrente consumida, fornecer a informação sobre qual eletrodoméstico está ligado à energia em determinado momento. Através dos experimentos, foi selecionada a melhor configuração para o tratamento dos dados, bem como a arquitetura da rede neural utilizada. O resultado obtido foi satisfatório para a maior parte dos eletrodomésticos testados, em que se conseguiu predizer com boa precisão qual o equipamento foi analisado.

Palavras-chave: inteligência artificial. *machine learning*. redes neurais. aprendizado de máquina. assinatura de corrente elétrica. transformada de fourier.

ABSTRACT

Electric energy consumption in the world has been changing progressively every year. Energy concessionaires have a recurring concern with meeting the increasingly growing demand, trying to avoid rationing, the use of thermoelectric plants, or even the popularly called “blackouts”. Such events are also reflected in concern on the part of the end consumer, since energy costs are high in these scenarios, and, in a worst case, services are interrupted. This work aims to study a practical implementation to bring information about which household appliances consume the most energy within a residential environment, through artificial intelligence. Another proposal is to also be able to offer information about the value of the electricity bill, even before being informed by the concessionaire. Together, this data can objectively provide control over more conscientious consumption by the population. For this implementation to be carried out, Machine Learning techniques were used, such as Artificial Neural Networks, to analyze the electrical current of household appliances. The feasibility of this implementation was validated according to the hypothesis of a characteristic in the current consumption of the devices, in which, due to their specific construction and internal circuitry, they have a pattern, peculiar to each one, in the waveform of the current consumed, which generates a kind of spectral “signature”, when analyzed in the frequency domain. Experiments and data collection were carried out using specific laboratory equipment connected to common household appliances, with the aim of training a neural network, so that it can, based on the current consumed, provide information about which appliance is connected to the energy at a given time. Through the experiments, the best configuration for processing the data was selected, as well as the architecture of the neural network used. The result obtained was satisfactory for most of the household appliances tested, in which it was possible to predict with good accuracy which equipment was analyzed.

Keywords: artificial intelligence. machine learning. neural networks. electrical current signature. fourier transform.

LISTA DE FIGURAS

LISTA DE ABREVIAÇÕES

CSV *Comma-separated Values.*

DDP Diferença de Potencial.

DFT *Discrete Fourier Transform.*

FFT *Fast Fourier Transform.*

IA Inteligênca Artificial.

IDE Ambiente de Desenvolvimento Integrado.

JS *Java Script.*

LED *Light Emitting Diode.*

ML *Machine Learning.*

MLP *Multi Layer Perceptron.*

ReLU *Rectified Linear Unit.*

RNAs Redes Neurais Artificiais.

SUMÁRIO

1 INTRODUÇÃO

A energia elétrica é uma das principais fontes de energia utilizadas pelo ser humano, estando relacionada ao crescente desenvolvimento das sociedades e ao aumento da qualidade de vida da população global. Ainda que o Brasil seja um país de grandes recursos naturais renováveis, e que consiga altas produções energéticas, ainda existem problemas quanto ao fornecimento de energia no país. Ademais, o custo associado ao consumo de energia elétrica se mostra como uma preocupação recorrente (ABRAHÃO; SOUZA, 2021) (STILPEN; CHENG, 2015). Seja por parte das fornecedoras de energia, ou por parte do consumidor final, saber o quanto se gasta de energia a qualquer momento pode ser bastante proveitoso. Mais do que isto, entender de onde vêm os maiores consumos é essencial para se ter uma utilização mais eficiente e mais inteligente da energia. Este trabalho tem como objetivo estudar técnicas de reconhecimento no padrão de consumo de corrente elétrica para a identificação de eletrodomésticos.

1.1 TEMA DO PROJETO

Este projeto visa estudar técnicas de reconhecimento de padrões no consumo de corrente elétrica por meio de algoritmos de Inteligência Artificial (IA).

1.2 PROBLEMATIZAÇÃO

Vivemos uma crise elétrica no Brasil, no qual existe certa dificuldade no fornecimento de energia por meio das usinas hidrelétricas, resultando no uso excessivo de termelétricas, como de gás ou carvão, ou mesmo a importação de países vizinhos para suprir a demanda energética. Este fornecimento de energia alternativo acaba por gerar problemas ambientais devido a emissão de poluentes, bem como problemas econômicos, visto os maiores gastos envolvendo termelétricas e importações. Para o consumidor final, o primeiro passo para um consumo mais eficiente é saber de onde vêm os maiores gastos. Neste sentido, diferenciar os eletrodomésticos quanto ao consumo de energia é de suma importância. No mercado nacional e internacional, existem soluções que são capazes de medir e mostrar o quanto cada eletrodoméstico consome de energia dentro de uma residência. No entanto, estes equipamentos são individuais e precisam ser ligados a cada aparelho eletroeletrônico separadamente para fazer a medição. Ademais, estes equipamentos são de difícil acesso por não possuírem um baixo custo. Neste sentido, para obtenção de informações de consumo sobre cada eletrodoméstico, seria necessário ter uma rede de vários medidores ligados simultaneamente, o que, além de ter um custo elevado, pode trazer um consumo de energia adicional considerável à residência. Diante deste cenário, observa-se a necessidade de um equipamento capaz de identificar quais eletrodomésticos estão ligados, e o quanto cada um consome, com um único *hardware* de medição.

1.3 HIPÓTESES

Os equipamentos elétricos residenciais compartilham, entre si, o consumo de tensão (110 ou 220V), em 60Hz no Brasil. Porém, cada um tem um jeito peculiar e específico de consumir corrente, devido ao arranjo dos seus componentes eletrônicos internos. Esta especificidade pode tornar cada tipo de aparelho eletrodoméstico único, com uma espécie de assinatura de corrente elétrica. Visto isso, a proposta então é validar, estudar e aplicar uma prática muito utilizada em telecomunicações, em um ambiente elétrico. Esta prática consiste em analisar e identificar diferentes sinais transmitidos em meio a um aglomerado de sinais sobrepostos utilizando ferramentas matemáticas.

1.4 OBJETIVOS

1.4.1 Objetivo Geral

Reconhecer o padrão de consumo de diferentes eletrodomésticos, quando ligados isoladamente ou em conjunto, para validar as hipóteses de que cada um possui uma assinatura de corrente, e que com ela, é possível identificá-los e diferenciá-los e que se consegue identificar no espectro de frequência, as componentes que identificam cada equipamento, podendo assim, com a análise do espectro, entender quais dispositivos estão ligados à rede.

1.4.2 Objetivos Específicos

Para a validação da assinatura de corrente:

- Encontrar um equipamento/dispositivo de medição que seja capaz de medir corrente elétrica com uma alta frequência de amostragem, para poder detectar as variações que possivelmente serão responsáveis pela assinatura de corrente;
- Medir diferentes aparelhos eletroeletrônicos, tanto da mesma categoria, quanto de categorias diferentes, para se realizar uma inspeção visual, e observar as diferenças gráficas entre o consumo de corrente deles;
- Padronizar a captação dos dados e tratá-los para que sirvam de entrada para algum algoritmo de IA;
- Buscar um algoritmo de IA capaz de agrupar as medições por características e similaridades.

Para a validação da identificação de vários eletrodomésticos ligados ao mesmo tempo com uma única informação do espectro:

- Converter cada medição isolada para o espectro de frequência utilizando o algoritmo de *Fast Fourier Transform* (FFT);
- Definir uma metodologia de padronização e tratamento dos dados para reconhecer as componentes mais importantes para a diferenciação na assinatura de um sinal;
- Realizar a medição de vários equipamentos ligados de forma simultânea e converter esta medição para o domínio da frequência por meio de FFT;
- Identificar as componentes que foram medidas de forma isolada no espectro que contém o aglomerado.

1.5 JUSTIFICATIVAS

Atualmente, a população ainda carece de meios eficazes e acessíveis para obter informações a respeito de seu consumo elétrico. Um sistema de medição capaz de reconhecer em tempo real o quanto cada eletrodoméstico consome em uma residência é uma forma de entregar à população uma ferramenta que lhe traga mais autonomia. Saber, em qualquer dia do mês, o quanto cada equipamento já consumiu e quanto este consumo afeta sua conta de energia permite que o consumidor otimize seu consumo de forma individual, resultando em economias mensais significativas.

Analisando coletivamente, este consumo de energia mais eficiente e inteligente por parte da população resulta em menores gastos energéticos no país e, consequentemente, a redução do uso de usinas termelétricas e importação de energia para território nacional.

Para que exista viabilidade, é importante que esse sistema seja de fácil implementação e baixo custo, o que impede a difusão e volume de utilização robusto das soluções já existentes atualmente.

Este trabalho visa o estudo de técnicas para o desenvolvimento de um sistema que consiga entregar a informação do consumo por eletrodoméstico utilizando apenas um equipamento em cada residência. De modo a reduzir os gastos agregados ao hardware, propõe-se ainda trabalhar com processamento em nuvem, permitindo com que o equipamento medidor tenha somente a responsabilidade de captar os dados, e não de processá-los.

1.6 CONSIDERAÇÕES FINAIS

A energia elétrica se tornou um recurso indispensável para o modo de vida atual. No Brasil, grande parte de sua produção é advinda de usinas hidrelétricas. No entanto, a falta de chuvas e consequentemente a baixa nos reservatórios resultaram em grandes crises energéticas a partir do ano 2000. A falta de conhecimento por parte da

população do quanto de energia se consome nas casas também impacta na situação supracitada. Como resultado, dois problemas são gerados: a conta de energia tem se tornado cada vez mais cara para o consumidor, e o risco de uma crise energética devido ao alto consumo e atividade reduzida das hidrelétricas se torna frequente.

Ter um modo de consultar, a qualquer momento, quanto cada equipamento já consumiu e o quanto isto impacta na conta de energia se mostra de grande valia. Neste sentido, é proposto um estudo de técnicas de reconhecimento de padrões no consumo de corrente elétrica por meio de algoritmos de IA.

No Capítulo 2 é apresentado o referencial teórico utilizado na elaboração do projeto, incluindo informações a respeito das características de consumo de corrente elétrica, e de algoritmos de inteligência artificial para aprendizado de máquina, que serão utilizados para identificação dos eletrodomésticos ativos em uma residência. No Capítulo 3 são apresentados os materiais e métodos do projeto, com a descrição de cada uma das etapas utilizadas para a implementação do mesmo. O Capítulo 4 apresenta os resultados obtidos após a implementação e, por fim, o Capítulo 5 mostra a conclusão do trabalho.

2 REFERENCIAL TEÓRICO

A matriz energética brasileira é notavelmente diversificada, sendo composta por uma variedade de fontes de energia, renováveis e não renováveis, sendo a maioria advinda do primeiro grupo, com destaque para as hidrelétricas. No ano de 2022, cerca de 58% de toda a produção energética brasileira foi proveniente de usinas hídricas. Quando avaliamos o perfil mundial, o Brasil se destaca, sendo o sétimo país com maior capacidade instalada de geração elétrica, e o sexto maior consumidor no ano de 2021 (EPE, 2023).

O crescente desenvolvimento do país permitiu o aumento na qualidade de vida e saúde dos brasileiros. Com maior poder de aquisição, a compra de eletrodomésticos aumentou por parte dos consumidores, cada vez mais adeptos ao uso de lava-louças, panelas elétricas, fornos elétricos e principalmente de ares-condicionados (ABRAHÃO; SOUZA, 2021). Como consequência, o consumo de energia elétrica se mostra em constante aumento conforme o passar dos anos.

Ainda que o país tenha uma alta capacidade instalada de geração elétrica, crises energéticas se tornaram frequentes, com a pior delas ocorrendo em 2001. Devido à escassez de chuvas, o potencial de produção por parte de usinas hidrelétricas se tornou reduzido. O chamado apagão de 2001 perdurou por cerca de 9 meses, indo de Maio de 2001 a Fevereiro de 2002, resultando em graves consequências políticas e principalmente econômicas. Outros casos semelhantes de crises energéticas no país foram registrados nas últimas duas décadas (BOLETINS. . . , 2016; MARTINS, 2021; MALAR, 2022; BORGES, 2021).

2.1 CARACTERÍSTICAS DO SINAL ELÉTRICO

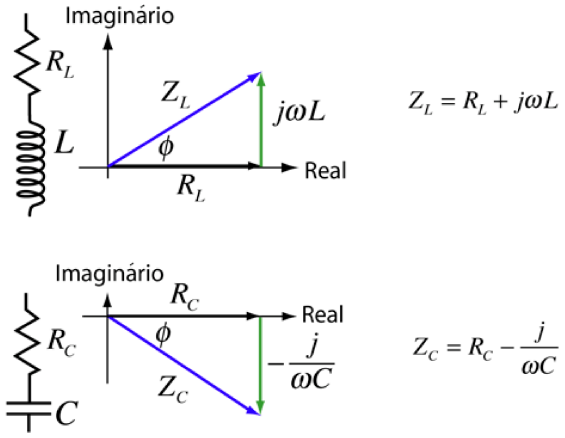
Todos os componentes elétricos ou eletrônicos têm uma característica em comum: para funcionar eles consomem tensão e corrente elétrica. A tensão consumida por um componente depende da diferença de potencial elétrico (DDP) à qual o mesmo foi submetido. Esta DDP pode ser oriunda de uma fonte de tensão, como uma bateria, ou uma tomada, bem como de um circuito associado em paralelo a este componente que forneça alguma energia a ele. Em todos os casos, a tensão de um componente depende de uma fonte que entrega energia a ele. A corrente elétrica instantânea consumida por um equipamento ou componente eletrônico tem seu valor dependente tanto da DDP aplicada a ele, quanto da impedância interna do mesmo. Tal impedância depende das características físicas do material de que o componente é feito. A impedância se manifesta de forma resistiva, reativa, ou das duas ao mesmo tempo, e cada componente elétrico/eletrônico possui uma impedância característica. Um componente bastante conhecido é o resistor. O resistor tem a capacidade de transformar corrente elétrica em calor, através do efeito *Joule*. Um resistor ideal possui uma impedância puramente resistiva, não apresentando nenhuma

parcela reativa. Outro componente bastante conhecido é o capacitor, um elemento que tem a capacidade de armazenar energia na forma de campo elétrico. Um capacitor ideal possui uma impedância puramente reativa, não apresentando nenhuma parcela resistiva. O mesmo acontece com um indutor, que tem a capacidade de armazenar energia na forma de campo magnético. O indutor ideal também não possui parcela resistiva em sua impedância, somente a parte reativa. Em termos práticos, a parte resistiva de qualquer impedância dissipa a energia na forma de calor, e a parte reativa causa um adiantamento ou atraso da corrente com relação à tensão. Tais comportamentos da parte reativa de uma impedância (também chamada de reatância) acontecem quando se tem uma fonte de corrente alternada, e a intensidade de cada efeito está associada não somente a esta reatância, mas também à frequência de oscilação da corrente elétrica (BOYLESTAD, 2012). Uma impedância Z pode ser representada matematicamente por um número complexo, de acordo com a Equação 1, em que R é a parte resistiva, e X a parte reativa.

$$Z = R + iX[\Omega] \tag{1}$$

Quando descrevemos uma impedância na forma fasorial, temos uma representação geométrica da parte real e da parte imaginária no plano cartesiano, o que nos dá a visualização do ângulo de defasagem, ângulo este que está associado ao adiantamento/atraso de fase citado anteriormente (BOYLESTAD, 2012). A Figura 1 representa esta relação.

Figura 1: Representações fasoriais da impedância.



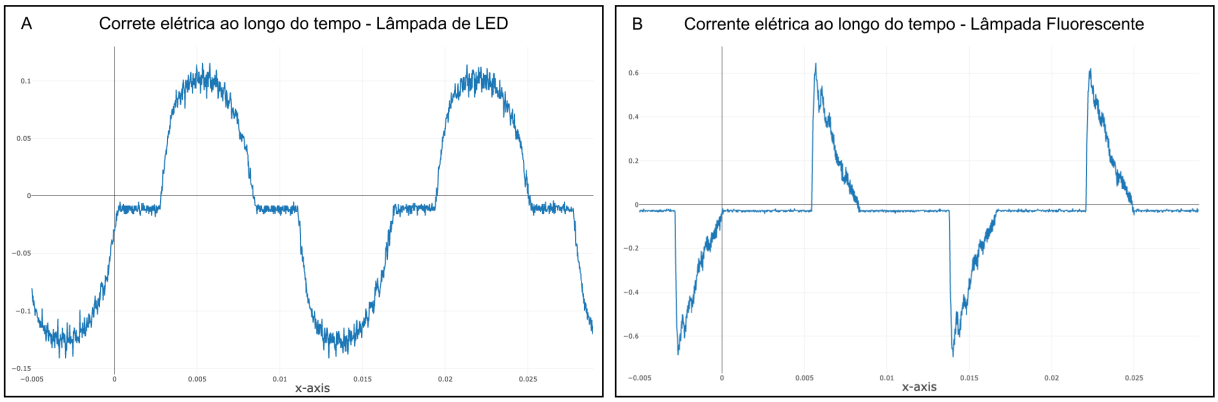
Fonte: Adaptado de (NAVE, 2000)

Os eletrodomésticos são construídos, na maioria das vezes, com a utilização de um conjunto de componentes arranjados de forma peculiar e específica. Além das variações provocadas na corrente pela impedância, existem ainda outras variações causadas por componentes não lineares, como é o caso da família dos semicondutores. Entre diodos,

transistores, *mosfets*, e vários outros, têm-se comportamentos de consumo de corrente específicos. Quando um emaranhado de componentes são arranjados para se ter um funcionamento específico de um eletrodoméstico, este conjunto produz um padrão de consumo de corrente estacionário que contém componentes de frequência específicas que podem ser utilizadas para se determinar uma assinatura daquele aparelho.

Pode-se observar, na Figura 2, exemplos de lâmpadas que possuem diferentes construções: uma LED, e a outra fluorescente. Cada uma possui um comportamento distinto quanto ao consumo de corrente.

Figura 2: Corrente ao longo do tempo para duas lâmpadas. Em A, lâmpada de LED. Em B, lâmpada fluorescente.



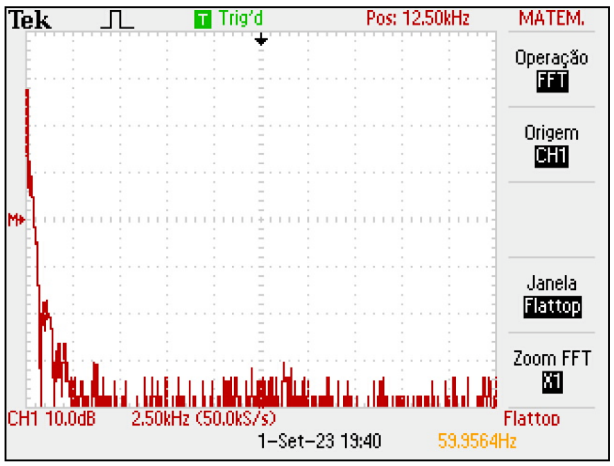
Fonte: O autor.

Um conceito amplamente utilizado em telecomunicações é o de análise de frequências. Quando se tem um sinal composto por uma soma de vários sinais sobrepostos, pode-se usar a transformada de Fourier para analisar as componentes em frequência do mesmo. Sabe-se que um eletrodoméstico que consome uma corrente que é composta por um conjunto de diversas frequências, pode ser analisado como um sinal de telecomunicações. Nele, podem ser usadas ferramentas matemáticas para melhor análise, como por exemplo a transformada de Fourier, que converte o sinal do domínio do tempo para o domínio da frequência. Com esta transformação, pode-se observar as principais frequências que estão presentes naquele consumo de corrente (OPPENHEIM A. V; WILLSKY, 2010).

Na Figura 3, pode-se observar o espectro de frequências da corrente de um carregador de *smartphone* ligado à rede elétrica de 60Hz, medido por um osciloscópio. Observa-se que existem altas frequências associadas a este consumo.

Vários eletrodomésticos diferentes, ligados ao mesmo tempo, produzirão na rede elétrica de entrada de uma residência um conjunto de todas as frequências utilizadas por eles. Assim, sabendo o padrão de consumo de cada um, pode-se identificar quais eletrodomésticos estão ativos em determinado momento, o consumo total de energia, e ainda discriminar o consumo individual de cada um.

Figura 3: Espectro de frequência da corrente de um carregador de *smartphone*.



Fonte: o autor

2.2 APRENDIZADO DE MÁQUINA

Para que se possa entender qual é o padrão de consumo de corrente de cada eletrodoméstico, serão utilizadas técnicas de IA. O aprendizado de máquina, ou *machine learning* (ML), é o campo de estudo da inteligência artificial que visa que um computador realize uma tarefa sem que se precise programá-lo diretamente para tal. Uma programação prévia é feita para que um algoritmo aprenda, sozinho, como deve fazer algo. As redes neurais artificiais (RNAs) são exemplos de algoritmos de *machine learning*. As RNAs são muito eficientes em casos em que se precisaria de um algoritmo convencional com muitos parâmetros, configurações e ajustes, como é o caso de sistemas muito complexos, como por exemplo o da aplicação presente neste trabalho (KOUTROUMBAS; THEODORIDIS, 2009).

O aprendizado de máquina tem por essência a forma como os seres humanos aprendem. Quando nascem, não têm praticamente nenhum parâmetro para analisar as coisas ao redor. Assim, não conseguem distinguir objetos, classificá-los em categorias, entender quais são perigosos ou não, entender emoções, estímulos visuais, auditivos, entre outros. À medida que o tempo passa, cada acontecimento é captado pelos sensores (visão, audição, tato, paladar e outros) e interpretado e armazenado pelo cérebro. Assim, depois de algum tempo, conseguem reconhecer padrões baseados em características similares às já vistas anteriormente. Conseguem identificar que um objeto é, por exemplo, um guarda-chuva, mesmo que não se tenha nunca visto aquele modelo específico anteriormente, devido às características comuns de vários outros guarda-chuvas vistos anteriormente. O aprendizado de máquina tem um funcionamento similar, que tenta associar características de entrada com alguma saída conhecida, ou então agrupar várias entradas com base em características semelhantes.

Existem alguns tipos de ML, como os supervisionados, os não supervisionados,

e os semi-supervisionados. Os supervisionados são aqueles em que se têm uma grande base de dados, e nela, cada conjunto de dados é rotulado, para que se possa ensinar ao sistema sobre o que é cada elemento. Por exemplo: pode-se ter uma base de dados que contenha fotos de veículos. Neste exemplo, imagina-se que se tenha apenas fotos de carros e motos. Todas as fotos de carros possuem uma etiqueta que contém a informação carro, e todas as fotos de motos, contêm a etiqueta moto. Serão inseridas como entrada para o sistema tanto as fotos quanto as respectivas etiquetas. Com uma base de dados grande o suficiente, depois de treinada, espera-se que esta IA supervisionada seja capaz de reconhecer os principais parâmetros que definem se a foto é de um carro ou de uma moto. Então, quando temos como entrada uma foto que pode ser de uma moto, ou de um carro, mas que não tenha a etiqueta identificadora, a IA produzirá uma saída com a informação esperada: é uma moto, ou então, é um carro. Quanto mais dados de treinamento, ou seja, quanto mais fotos, mais capaz a IA será de distinguir as duas categorias.

Os não-supervisionados, por sua vez, são aqueles em que não se atribui uma etiqueta sobre cada entrada dos dados de treinamento. Logo, a IA "não sabe" qual elemento está sendo utilizado como entrada, mas, ainda sim, deve ter a capacidade de agrupar conjuntos de dados em características similares (KOUTROUMBAS; THEODORIDIS, 2009). Os semi-supervisionados possuem características tanto dos supervisionados quanto dos não supervisionados. Estes podem usar uma base de dados de aprendizado rotulada não muito extensa, e trabalhar com entradas não rotuladas, aliadas ao conceito de reforço, em que uma intervenção humana auxilia a IA a categorizar se a saída produzida foi correta ou não.

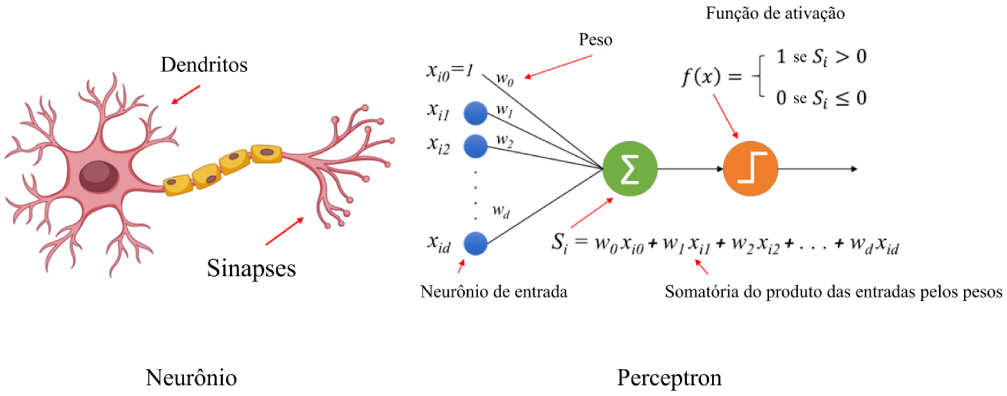
2.3 REDES NEURAIS ARTIFICIAIS

A rede neural artificial, ou somente rede neural, é uma técnica de ML que visa resolver um problema qualquer baseado no funcionamento do nosso cérebro. Nem sempre usar redes neurais é a melhor forma de resolver um problema, mas funciona bem para casos em que não se têm uma regra muito bem estabelecida sobre o funcionamento do objeto de estudo em questão, como por exemplo detecção de caligrafia, ou reconhecimento facial (GRUS, 2016).

Nosso cérebro é constituído por uma rede de neurônios conectados entre si, e estes recebem estímulos uns dos outros, e propagam impulsos nervosos. Para que um neurônio dispare um impulso nervoso, uma condição específica dentro dele precisa ocorrer, e esta condição é baseada nos estímulos vindos de outros neurônios conectados a ele. Simplificadamente, é como se acontecesse dentro do neurônio um cálculo matemático que pondera as entradas, e se estas forem de tal forma que se atinja um "limiar de disparo" específico, o neurônio dispara o impulso nervoso. Na prática, o que ocorre são reações químicas muito peculiares, que fazem o nosso cérebro como um todo funcionar.

A topologia mais simples de rede neural artificial é a *perceptron*, e ela se aproxima do funcionamento de um único neurônio, em que o mesmo "enxerga" entradas, atribui pesos às mesmas, e realiza um cálculo matemático. Se determinada condição foi satisfeita, a rede "dispara" um sinal de saída (GRUS, 2016). Pode-se ver na Figura 4 o esboço de um *perceptron*.

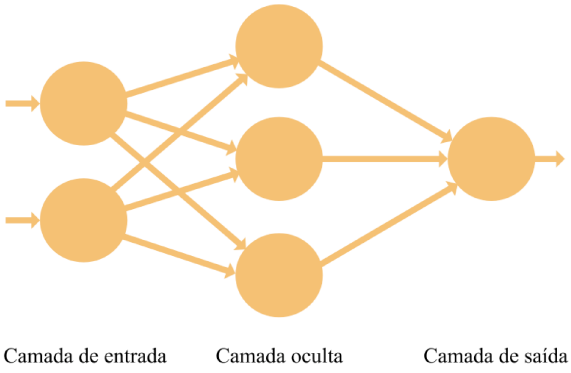
Figura 4: Neurônio e *perceptron*.



Fonte: Adaptado de (GUZMAN, 2019)

Outra topologia comumente utilizada é a *Multi Layer Perceptron* (MLP), em que são atribuídas camadas intermediárias à saída e às entradas. Estas são chamadas de camadas ocultas. As MLPs surgiram para resolver problemas mais complexos em que apenas um *perceptron* não é capaz de solucionar. Neste modelo de rede neural, os neurônios das camadas intermediárias conectam-se com todos os elementos vizinhos, como pode ser visto na Figura 5.

Figura 5: RNA *Multi Layer Perceptron*.



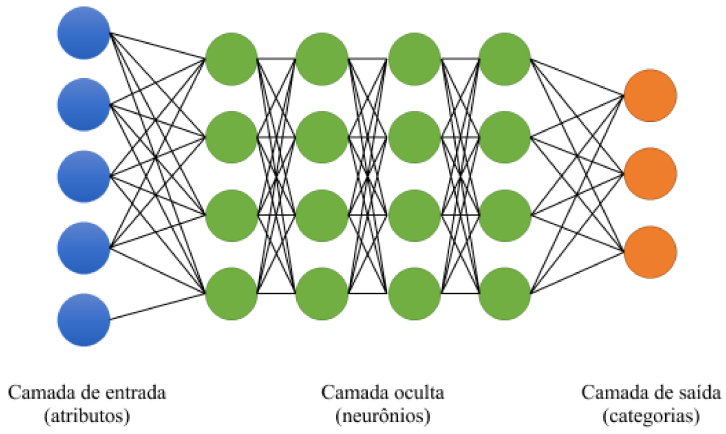
Fonte: Adaptado de (BERGER, 2016)

Em uma MLP, os neurônios da camada de entrada estão associadas aos parâmetros do problema a ser resolvido. São eles que impactarão na saída da rede neural.

No exemplo supracitado da aplicação que visa diferenciar e classificar imagens de carros e motos, estes parâmetros de entrada poderiam ser, por exemplo: peso, tamanho, cor, proporção, entre vários outros. Para o caso de reconhecimento de imagens, pode-se utilizar técnicas de pré-processamento para extrair informações otimizadas, como por exemplo distância entre elementos principais (distância entre as rodas por exemplo), bordas, formas geométricas especificas entre outros. Existe uma gama de algoritmos especializados em pré-processamento, como é o caso da função de *Canny*, para detecção de bordas, ou mesmo o método de *Hough*, que é usado para detecção de formas facilmente parametrizadas, como linhas, círculos, elipses, entre outros.

Outra técnica utilizada é a de redes neurais convolucionais, que visa aplicar os filtros de pré-processamento de uma forma automática e otimizada, o que será melhor detalhado posteriormente. A camada de saída de uma RNA pode estar associada a um resultado quantitativo ou qualitativo, dependendo da forma como se projetou a rede. Ela pode por exemplo resultar em um valor numérico entre 0 e 1, ou entre -1 e 1. Desta forma, atribui-se esta saída a um resultado esperado, como por exemplo: quanto mais próximo o valor for de -1, classifica-se como moto, ou quanto mais próximo o valor for de 1, classifica-se como carro. Podem haver um ou vários neurônios na camada de saída. A camada oculta pode conter várias camadas intermediárias, como pode ser visto na Figura 6, sendo estas responsáveis por fazer conexões entre si, de forma a relacionar de diferentes maneiras os parâmetros de entrada.

Figura 6: RNA *Multi Layer Perceptron* com várias camadas.

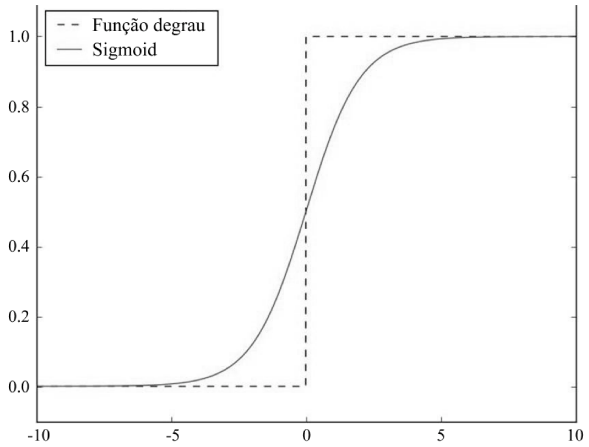


Fonte: Adaptado de (GUZMAN, 2019)

A maior parte das redes neurais funcionam como uma “caixa preta”, em que a ação de analisar os detalhes internos não nos fornece muito entendimento sobre como o problema está sendo resolvido (GRUS, 2016). Na prática, um neurônio processa as informações vindas dos elementos anteriores por meio de uma função de ativação, que é uma equação matemática que visa transformar a entrada, ou a soma de várias entradas,

numa saída. A função de ativação pode ser de várias formas, porém existem algumas mais utilizadas, como por exemplo a função degrau, ou então a função *sigmoid*, que pode ser vista como uma espécie de degrau suavizada. Em termos de custo computacional, a *sigmoid* é uma melhor opção, visto que não tem descontinuidades como a degrau, como pode ser observado na Figura 7 (GRUS, 2016).

Figura 7: Função degrau e função *sigmoid*.



Fonte: Adaptado de (GRUS, 2016)

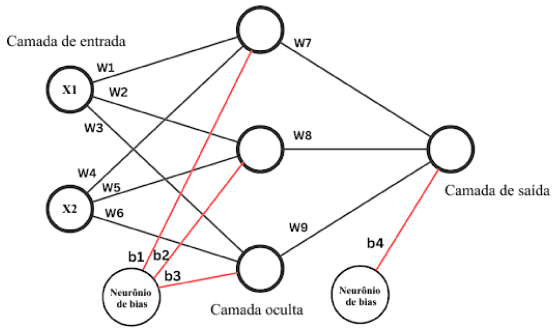
A função de ativação mapeia uma entrada em uma saída. Tanto a degrau quanto a *sigmoid* funcionam de forma a tentar limiarizar um parâmetro, comparado ao que acontece em uma estrutura de programação "*if else*", ou seja, uma decisão, porém, uma decisão que não é discreta, e sim contínua, podendo ser interpretada como uma tendência, ou mesmo uma saída que é mais próxima, ou mais distante de um resultado. Pode-se entender como: se o valor de entrada está abaixo ou acima de um certo limiar, a saída é muito próxima de um extremo, próximo do comportamento de um "*if else*", como pode ser observado na Figura 7. Existem outros tipos de função de ativação, que podem ser úteis em casos específicos, como por exemplo a função *softmax*, que é utilizada para casos em que se trabalha na classificação de mais de dois tipos de categorias, e também a função *ReLU*, que é uma abreviação para *Rectified Linear Unit*, e seu funcionamento consiste em se retornar zero para todos os valores negativos, e uma resposta linear para valores positivos, sendo o valor de saída igual ao de entrada.

A rede neural MLP como um todo tem seus neurônios ligados entre si, como pode ser observado na Figura 6. Estas ligações possuem diferentes intensidades. Em outras palavras, a informação de um parâmetro de entrada pode ter mais peso sendo propagada por um determinado caminho do que por outro. Esta diferença de pesos dos caminhos faz com que a rede consiga atribuir diferentes resultados de saída para diferentes valores de entrada, possibilitando ainda que diferentes combinações dos valores de entradas se relacionem de diferentes formas ao se propagar pela rede neural. Assim,

quando ajustada de forma suficientemente adequada, pode resolver um problema. Este ajuste consiste basicamente no ato de alterar os pesos dos caminhos para que se chegue em uma configuração eficiente.

O processo de treinamento, ou aprendizado, de uma rede neural acontece pela mudança intencional dos pesos concomitantemente à análise das saídas, que são comparadas às respostas conhecidas e esperadas para determinadas entradas, para que se possa reforçar certa configuração de pesos e seguir com ajustes finos, ou então, mudar a direção da configuração dos pesos. Além dos pesos dos caminhos que ligam neurônios, existem também pesos associados aos *bias*, que são constantes adicionadas a cada neurônio, podendo ter o mesmo valor para uma camada inteira, ou então valores diferentes para cada neurônio. A representação dos pesos e dos *bias* pode ser observada na Figura 8.

Figura 8: Pesos e *bias* de uma rede MLP.

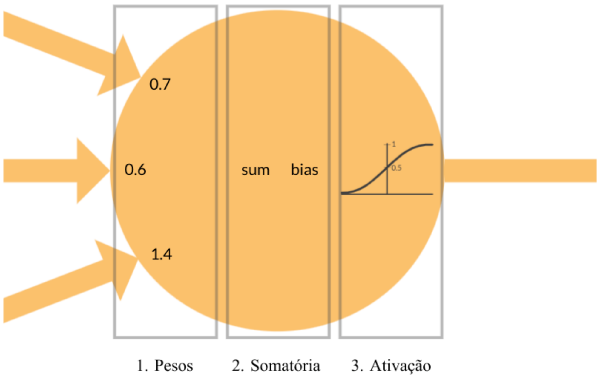


Fonte: Adaptado de (SIDHARTH, 2023)

As informações que são "entregues" a um neurônio, que não seja da camada de entrada, são as informações vindas das saídas de camadas anteriores, multiplicadas pelos pesos dos caminhos que elas percorreram. Logo, a função do neurônio é somar estas informações já ponderadas pelos seus pesos, acrescentar o valor do *bias*, e "passar" o valor resultante pela função de ativação, produzindo assim uma saída, como pode ser observado na Figura 9.

Existem algumas diferentes técnicas de se treinar uma rede neural. A que será usada neste trabalho é chamada de *backpropagation*, e visa corrigir os pesos em pequenos passos para que se tenha um erro cada vez menor. Este processo é aplicado na etapa em que se fornece uma entrada, e também o resultado esperado para ela, ou seja, um dado rotulado. A correção dos erros se inicia na camada de saída, e é propagada para as camadas ocultas anteriores, até se chegar na camada de entrada. O *backpropagation* usa várias relações de entrada e saída catalogadas para ajustar os pesos, e este processo faz com que o ajuste que diminui o erro para determinada entrada possa aumentar o erro de uma entrada diferente. É preciso se ter uma base de dados suficientemente grande para que a rede tenha informação para relacionar várias entradas com várias saídas e assim

Figura 9: Função do neurônio.



Fonte: Adaptado de (BERGER, 2016)

corrigir cada vez mais os pesos, e ter uma taxa de acerto adequada para o problema a ser resolvido.

2.4 BACKPROPAGATION

O processo de *backpropagation* é um algoritmo que tem como objetivo ajustar os pesos de uma rede neural com base no valor do erro produzido na saída, devido a uma determinada entrada. Uma rede neural é geralmente representada por um diagrama que relaciona visualmente as camadas entre si. Porém, na prática, o que um computador realiza é uma série de funções, onde várias camadas sucessivas são várias funções encadeadas. Logo, é possível minimizar o erro produzido por uma determinada entrada uma vez que conhecemos as funções que produziram certa saída e seu erro correspondente. Como o erro é a diferença entre um valor esperado e um valor resultante, o conceito de derivada local pode nos ajudar a encontrar as constantes multiplicativas das funções que irão fazer o erro tender a zero, uma vez que a derivada em um ponto representa a taxa de variação da grandeza analisada. Para a sucessão de funções encadeadas, são realizadas derivadas sobre derivadas para se obter cada um dos elementos multiplicativos que presentam os pesos das conexões da rede. Na camada de saída, uma única derivada da função no ponto pode resultar no valor que deve ser aplicado no peso desta última camada para fazer com que o erro tenda a zero. Na penúltima camada, é realizada a derivada da derivada sobre a função da penúltima camada que está dentro da função da última camada. Cada elemento da camada anterior está inserido dentro da função da próxima camada, assim, são realizadas derivadas sucessivas para que se possa corrigir os erros (LEITE, 2018).

Existem várias maneiras de se interpretar o erro associado. Na prática, têm-se alguns tipos de funções de perda, onde são atribuídos diferentes pesos para cada tipo de situação envolvendo o erro, como é o exemplo do erro médio quadrático, que é um dos

mais comuns, ou então o erro absoluto médio. No erro médio quadrático por exemplo, quanto maior for o erro absoluto, ou seja, a diferença entre o valor encontrado e o valor esperado, maior será o resultado da função de perda. Em outras palavras, a função de perda do tipo erro médio quadrático acentua os erros grandes, o que pode ser útil para que o algoritmo de *backpropagation* realize os ajustes de forma mais rápida. Outro tipo de função de perda é a de entropia cruzada, que é comumente utilizada quando se tem mais de duas categorias a serem classificadas, e sua principal característica é também acentuar erros grandes, porém de forma exponencial, e fazer com que erros associados a resultados não esperados tenham menos peso ao serem considerados. Este tipo de função de perda é geralmente utilizado em conjunto com funções de ativação do tipo *softmax*.

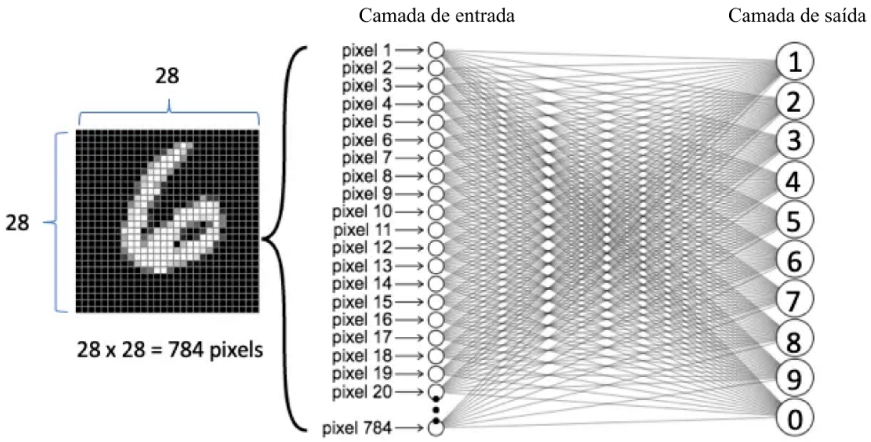
O algoritmo de *backpropagation* é o elemento responsável por fazer a rede neural "aprender". É um processo iterativo, onde o mesmo é realizado diversas vezes, e que, em cada iteração, o erro tende a diminuir. O número de vezes em que este algoritmo é realizado é chamado de épocas. Em cada época é executado o *backpropagation* para todos os dados de treinamento. O objetivo de se ter várias épocas é fazer com que o erro tenda a diminuir cada vez mais, pois, como citado anteriormente, o ajuste dos pesos que pode ser muito eficiente para uma determinada entrada, pode não ser tão bom para outra. Assim, depois de se realizar os ajustes para todas as entradas, uma nova iteração é realizada, repetindo-se todo o processo de *backpropagation*, fazendo com que o erro geral tenda a ficar menor. Contudo, existe um limite de ajuste, em que, mesmo se aumentando significativamente o número de épocas, o erro não diminui, ficando assim então a cargo de se aumentar o número de dados de treinamento para que se possa minimizar ainda mais o erro. Depois dos pesos ajustados de forma a se ter as menores perdas para todas as entradas de treinamento, o próximo passo é introduzir na rede neural ajustada, um dado não rotulado, e que tenha características semelhantes à algum elemento participante do grupo de treinamento, ou seja, um elemento pertencente a alguma categoria na qual a RNA foi treinada para classificar. Assim, a rede deverá se comportar de forma similar àquele caso ocorrido durante o treinamento, classificando o elemento não conhecido, atribuindo a ele um rótulo de saída condizente, uma vez que os pesos já foram ajustados para aquele tipo de situação. Pode-se dizer assim, que a rede neural "aprendeu".

2.5 REDES NEURAIS CONVOLUCIONAIS

Uma rede neural convolucional tem o objetivo de estratificar informações relevantes e quebrar o processo de classificação em etapas. Em um exemplo comum, que é o de reconhecimento de imagens, poderia ser utilizada uma rede neural MLP para classificar uma imagem, como pode ser visto na Figura 10. Vemos que, para cada *pixel*, temos uma informação associada, que neste caso é o nível da escala de cinza do *pixel*,

entre 0 a 255, em que 0 representa preto, e 255 branco. Desta forma, se treinássemos uma rede neural com muitas imagens do número 6, a rede poderia ser capaz de classificar uma nova entrada de um 6 não treinado. Porém, este tipo de aplicação está limitado às características espaciais da imagem, como por exemplo a posição em que o número foi desenhado. Aspectos também como a caligrafia da pessoa que escreveu, que pode tender a utilizar bordas mais arredondadas, ou mais quadradas, poderiam atrapalhar o processo de classificação, ou então, seria necessário uma quantidade muito grande de imagens do mesmo número, contendo muitas variações para se ter uma rede eficiente. A rede neural convolucional, por outro lado, é capaz de extrair informações relevantes da matriz do dado original que não dependam da posição em que foi desenhado, da caligrafia, da intensidade do nível de cinza, entre outras informações. Informações mais abstratas como forma, contorno, textura, ou até mesmo uma classificação local dentro de um objeto contido em uma imagem podem ser tipos de informações obtidas em um processo de convolução de uma rede neural. Este processo é feito relacionando a informação de um *pixel* com os *pixels* da sua vizinhança, por meio de convoluções da matriz original com uma matriz modificadora (GÉRON, 2019).

Figura 10: Rede Neural MLP utilizada para reconhecimento de imagem.

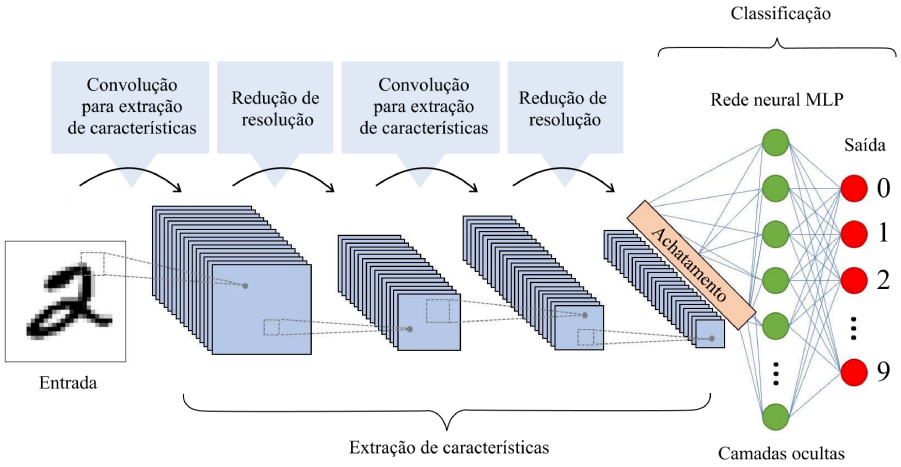


Fonte: Adaptado de (KUO, 2018)

Como citado anteriormente, seria possível também extrair características utilizando técnicas de processamento de sinais já conhecidas, como por exemplo, o filtro de Canny, que pode ser utilizado para a detecção de bordas, ou então o filtro de Hough, para detectar retas, ou círculos. Estas etapas de pré-processamento poderiam auxiliar as etapas seguintes de classificação de imagens. Nestes tipos de filtros, já se utiliza a convolução de uma matriz modificadora que é percorrida sobre a matriz original, realizando uma sucessão de convoluções para se chegar em um resultado. A ideia por trás das redes neurais convolucionais é fazer o que se é feito nos filtros utilizados na parte de pré-processamento, porém, ao invés de já se conhecer os parâmetros ótimos da matriz modificadora, numa

rede convolucional não se conhecem estes parâmetros, e não se sabe inicialmente qual é a melhor definição do filtro, então, a própria rede neural corrigirá e ajustará com o seu treinamento, encontrando assim os parâmetros otimizados para a aplicação em questão, de forma similar ao que é feito nas redes MLP. A etapa responsável por realizar as convoluções é chamada de *Feature extraction* (Extração de características). Esta etapa produz saídas que geralmente são ligadas a uma rede MLP convencional, que é nomeada de etapa de Classificação. A Figura 11 mostra um diagrama de uma rede neural convolucional.

Figura 11: Rede Neural Convolucional.



Fonte: Adaptado de (SAHA, 2018)

O conceito de rede neural convolucional parte do princípio de convolução, que é uma operação matemática realizada entre duas funções. Na prática, quando se tem uma matriz, ou mesmo um vetor de informações, pode-se realizar diversas convoluções entre o vetor original e um segundo vetor, repetindo-se este processo para várias localidades internas do vetor original. O tamanho da quantidade de posições que serão definidas para se realizar a convolução é chamado de *kernel*. Por exemplo, em uma matriz 9x9, poderia ser realizada uma convolução com alguns de seus elementos internos que formam uma matriz menor 3x3 com outra matriz externa 3x3. Esta matriz externa pode conter consigo elementos de uma função que modifica os valores da matriz original. Em um processo de convolução de uma rede neural convolucional, o *kernel* "percorre" a matriz original passando por todas as suas posições e gerando resultados destas operações, que são mapeados para uma nova matriz chamada de mapa de características, ou então *Feature Map*, e nela existirá o resultado das convoluções que foram percorridas na matriz original pela matriz modificadora, que é chamada de campo receptivo.

Quando este processo de convolução é realizado, pode-se dizer que o sistema está extraindo as informações mais relevantes de uma matriz original e passando para uma nova matriz. Assim, geralmente se tem uma menor quantidade de dados para repre-

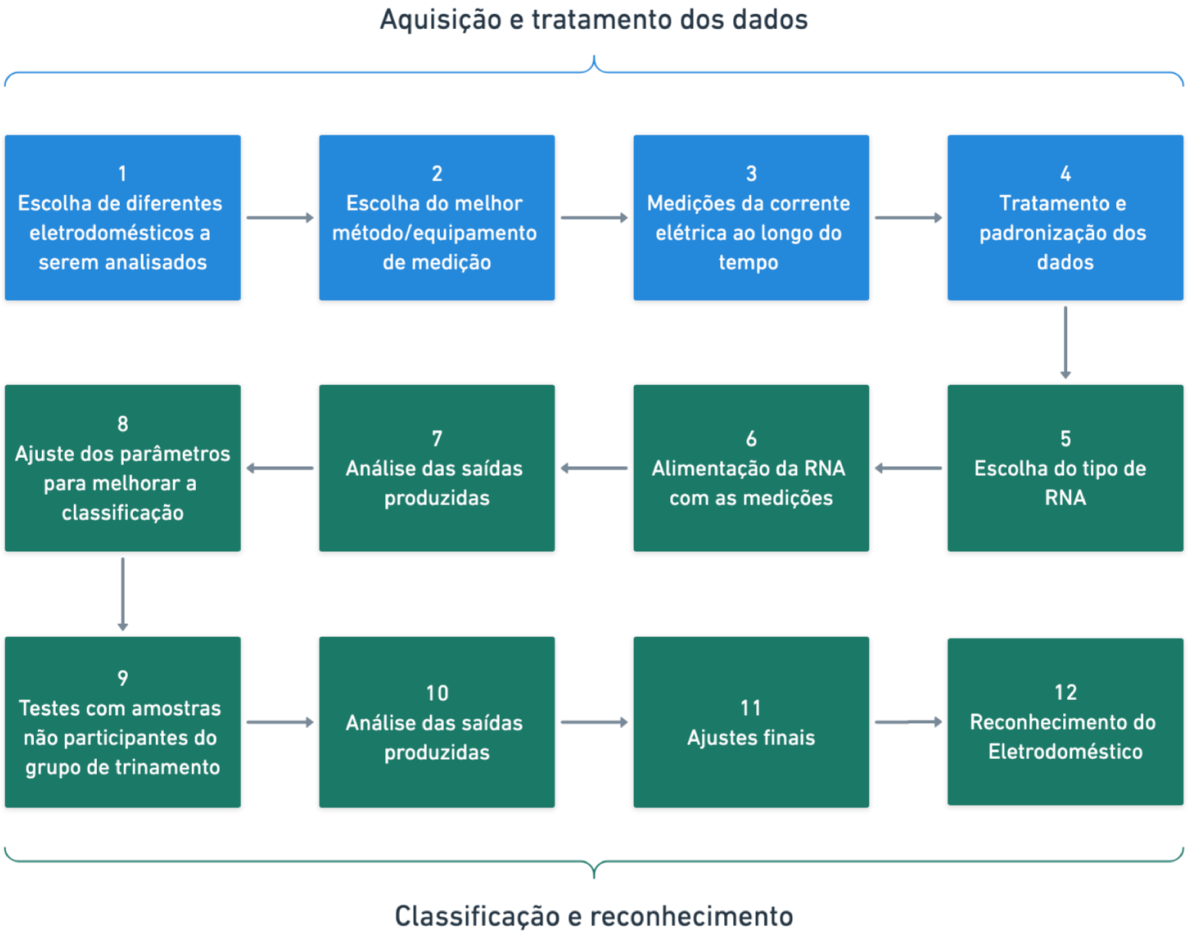
sentar informações mais relevantes. Por exemplo, em um caso de classificação de números escritos, caso tivéssemos como entrada o número 1, ao invés de termos como entrada as informações de vários *pixels* como a posição e a intensidade do nível de cinza de cada um, poderia existir uma única informação estratificada indicando que há uma borda na imagem que pode ser representada por uma linha vertical. Esta informação pode ser mais útil no processo de classificação das camadas posteriores, quando comparada ao caso dos *pixels* isolados, alcançando-se assim mais facilmente o resultado esperado.

Pode-se ter várias camadas convolucionais, onde existem diferentes matrizes de campo receptivo, e o objetivo destas várias camadas é que cada uma tenha o papel de extrair informações diferentes do sinal original. Por exemplo, em um caso de reconhecimento de imagem, uma camada pode ser responsável por extrair as bordas das imagens, e outro o gradiente de cores, etc. Estas informações podem ser usadas com suas respectivas importâncias na rede neural. Estas camadas também podem ser chamadas de canais. Também é comum existirem várias camadas convolucionais consecutivas, como pode ser visto na Figura 11, assim são extraídas cada vez mais informações estratificadas em cada camada. No exemplo citado acima, uma primeira camada convolucional poderia identificar bordas. A segunda poderia identificar se esta borda representa uma linha, ou uma curva. Uma terceira poderia "entender" se a linha é vertical ou horizontal, e assim por diante. Na prática não se tem a informação concreta do resultado destas convoluções, e, provavelmente não seríamos capazes de "decodificar" o resultado da convolução em algo concreto, mas em uma rede neural como um todo, os ajustes dos filtros das camadas convolucionais que são realizados no processo de aprendizagem faz com que a rede se adapte da melhor forma para entregar um melhor resultado.

3 MATERIAIS E MÉTODOS

Para a implementação deste projeto, foram realizadas duas grandes etapas: a de aquisição dos dados, e a de reconhecimento de padrão com redes neurais. Na etapa de aquisição dos dados, existe também a parte de tratamento e padronização. O diagrama da Figura 12 mostra uma visão geral da sequência realizada.

Figura 12: Diagrama visual das etapas para implementação do projeto.



Fonte: O autor.

Na etapa 1, escolheram-se grupos de equipamentos que provavelmente teriam comportamentos distintos quanto ao consumo de corrente elétrica. Dentro de cada grupo, escolheram-se equipamentos que provavelmente teriam comportamentos semelhantes, ou próximos, por terem uma mesma função. Esta etapa visa validar de forma visual tais comportamentos, de forma a tentar identificar padrões a olho nu, entre equipamentos semelhantes, e também conseguir diferenciar equipamentos de funções distintas. Neste sentido, a escolha dos eletroeletrônicos a serem analisados é de suma importância. Foram escolhidos os seguintes equipamentos, todos de posse do autor, representados na Figura 13:

- Lâmpada de LED da marca *Ourolux* (Figura 13A);
- Lâmpada de LED da marca *Lorenzetti* (Figura 13B);
- Lâmpada de LED da marca *Kian* (Figura 13C);
- Lâmpada de LED da marca *Avant* (Figura 13D);
- Lâmpada fluorescente da marca *Phillips* (Figura 13E);
- Lâmpada fluorescente da marca *Lorenzetti* (Figura 13F);
- Lâmpada fluorescente da marca *General Eletrics* (Figura 13G);
- Lâmpada fluorescente da marca *FLC* (Figura 13H);
- Carregador de *notebook* da marca *Dell* (Figura 13I);
- Carregador de *smartphone* da marca *Intelbras* (Figura 13J);
- Carregador de *smartphone* da marca *Samsung* (Figura 13K);
- Carregador de *smartphone* genérico, sem marca (Figura 13L).

A escolha dos equipamentos foi feita de forma arbitrária, porém com uma pressuposição de que o circuito das lâmpadas de LED se comportariam de forma diferente às dos circuitos das lâmpadas fluorescentes. De forma análoga, tem-se a pressuposição de que os 4 carregadores teriam um comportamento semelhante entre si, mas ainda com características peculiares de cada um. Tais suposições foram empiricamente analisadas por uma inspeção visual, o que será abordado nos passos 2 e 3.

Na etapa 2, foi feita a escolha do instrumento para medir os sinais elétricos. Tal escolha teve como ponto de partida a frequência de amostragem (F_s) mínima para aquisição dos dados. Esta F_s deve ter uma ordem de grandeza que obedeça ao teorema de *Nyquist*, em que F_s deve ser pelo menos duas vezes maior do que a máxima frequência do sinal amostrado. Como as variações que a corrente elétrica tem entre diferentes equipamentos acontecem dentro de um período de 60Hz, que é a frequência da rede elétrica no Brasil, sabe-se que dentro deste período, podem haver variações com frequências significativamente mais altas. Não se sabe qual é a maior frequência de oscilação dos sinais a serem analisados, então, o instrumento escolhido deve ter a maior F_s possível, e que seja várias vezes maior do que 60Hz. Quanto maior for a frequência de amostragem, mais resolução se terá do sinal analisado. Além desta característica, outro parâmetro para a escolha do instrumento de medição é que o mesmo tenha algum mecanismo de exportação dos dados, para que, além da análise visual, seja possível coletar as amostras para serem processadas pelos algoritmos de classificação.

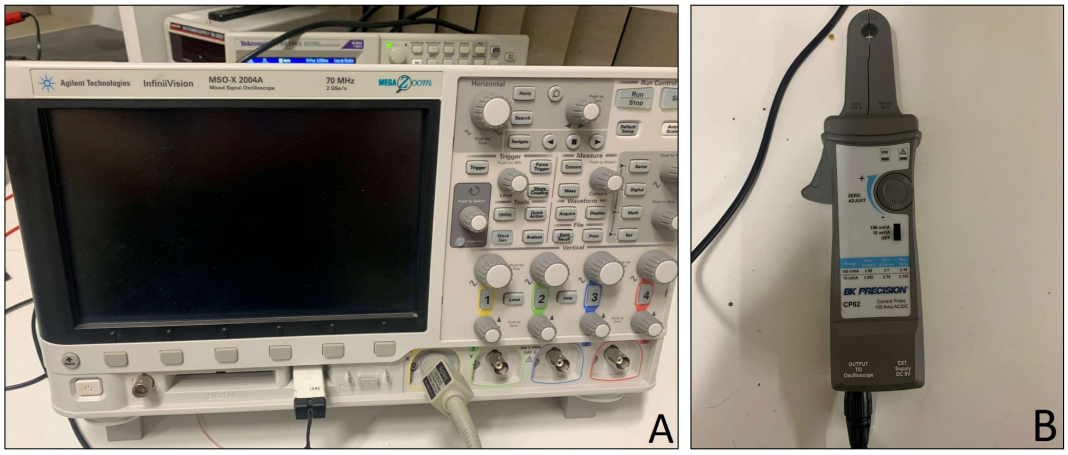
Figura 13: Aparelhos eletroeletrônicos selecionados para o experimento.

Fonte: O autor.

Escolheu-se um osciloscópio de 70MHz de frequência de amostragem para captar os sinais, o que atende ao critério estabelecido no parágrafo anterior. Este instrumento pode ser visto na Figura 14A. O osciloscópio conta com uma ponteira própria para medição de corrente, que funciona por indução e tem a sensibilidade da medição ajustável, podendo ser configurado em 10mV/A ou 100mV/A (Figura 14B). Como os equipamentos escolhidos no tópico anterior não têm um consumo de corrente elevado, configurou-se a ponteira para uma sensibilidade de 100mV/A. O osciloscópio possui também um recurso para exportar os dados medidos através de uma porta USB.

No passo 3 foram feitas as medições de cada equipamento, que resultaram nas

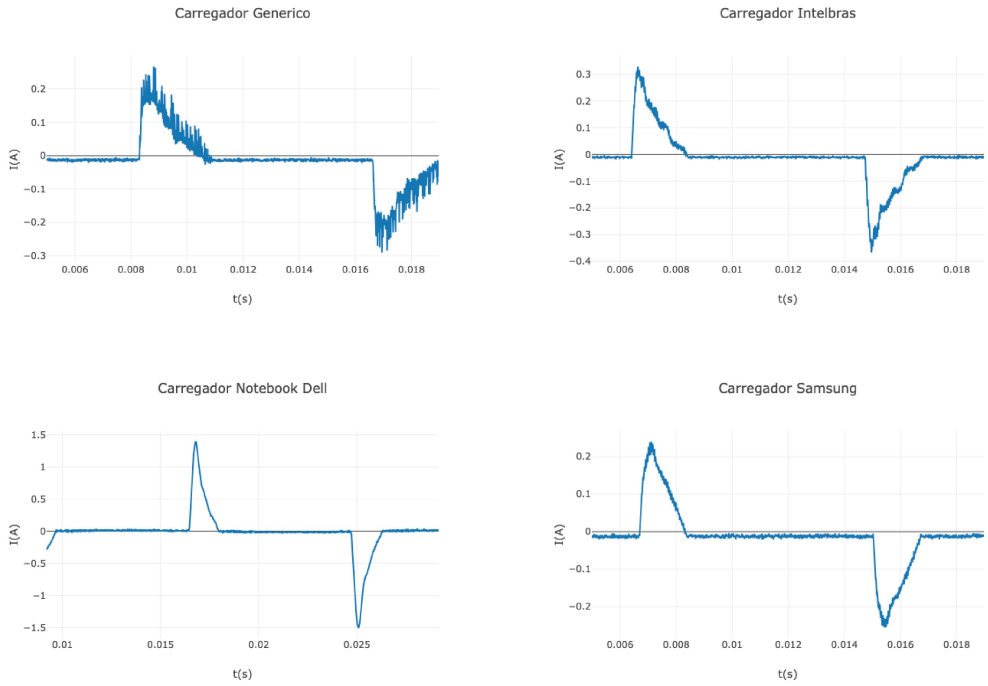
Figura 14: Instrumentos utilizados para medição de corrente elétrica. Em A, o osciloscópio. Em B, ponteira própria do equipamento utilizada para as medições.



Fonte: O autor.

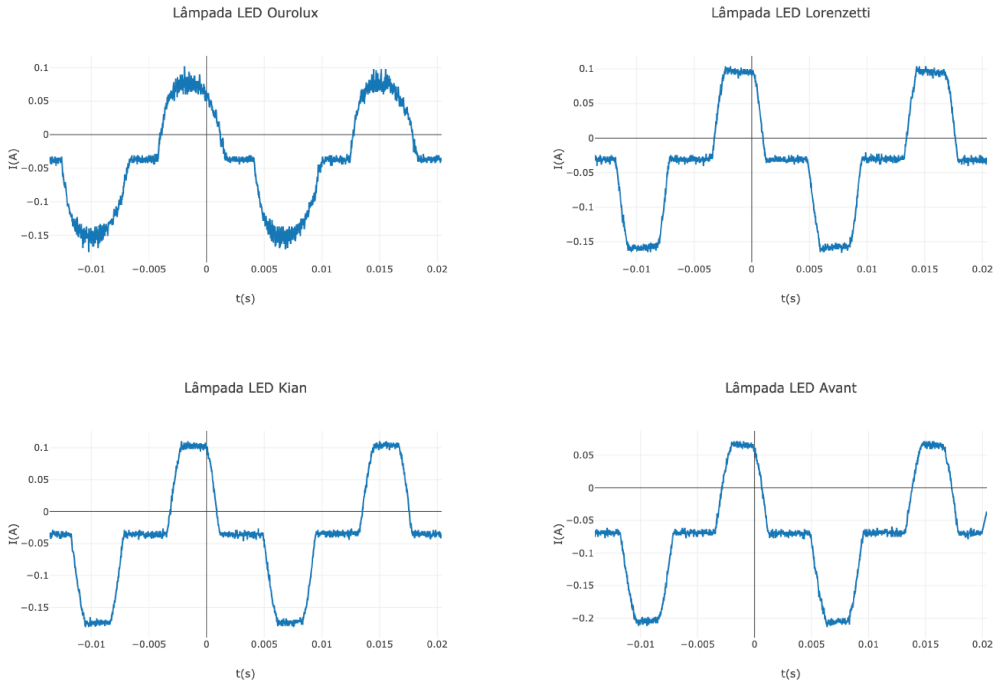
formas de onda apresentadas nas Figuras 15, 16 e 17.

Figura 15: Corrente elétrica ao longo do tempo para quatro carregadores.



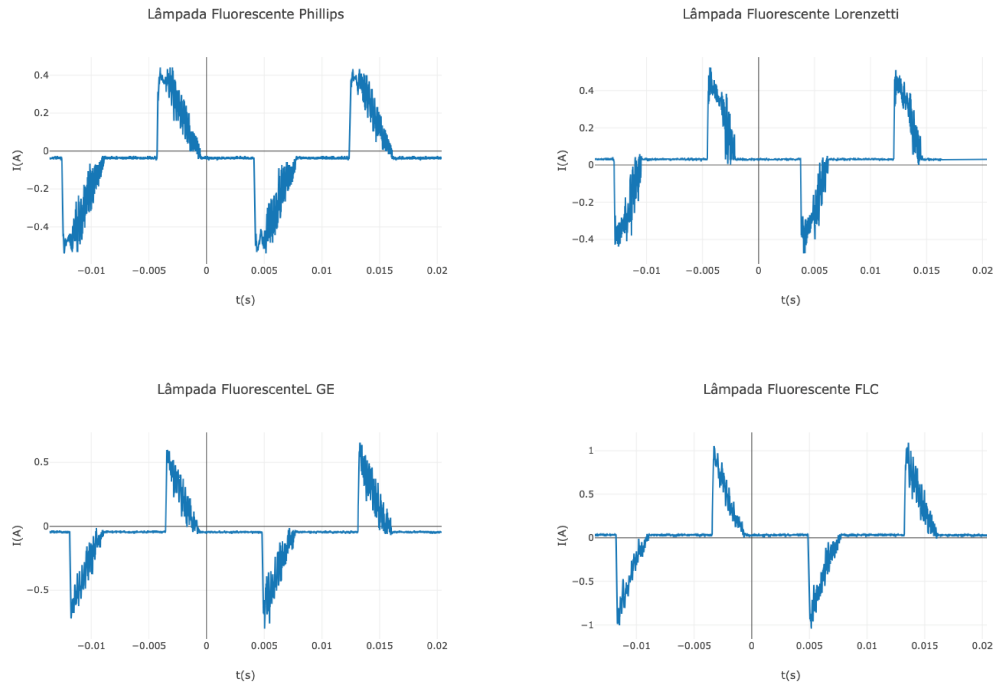
Fonte: O autor.

Figura 16: Corrente elétrica ao longo do tempo para quatro lâmpadas de LED.



Fonte: O autor.

Figura 17: Corrente elétrica ao longo do tempo para quatro lâmpadas fluorescentes.



Fonte: O autor.

3.1 RECURSOS

3.1.1 INSTRUMENTOS

Todos os instrumentos de medição utilizados foram fornecidos pelo Laboratório de Eletrônica do curso de Engenharia Eletrônica e de Telecomunicações da Universidade Federal de Uberlândia - Campus Patos de Minas.

O modelo do osciloscópio utilizado é o Keysight MSOX2004A de 70MHz, ilustrado na Figura 14A. Utilizou-se para a leitura, uma ponteira CP62 da BK Precision, adaptada para medição de corrente elétrica por indução, como visto na Figura 14B.

3.1.2 LINGUAGEM JAVASCRIPT

Escolheu-se a linguagem *Java Script* (JS) juntamente com seu ambiente de execução *Node.js*, por serem bastante difundidos e por se ter uma grande comunidade de utilização e apoio, além da familiaridade com a linguagem por parte do autor, o que traria agilidade e melhor entendimento com a parte técnica da implementação. A disponibilidade de uma biblioteca de *Machine Learning* para esta linguagem também foi um critério de escolha, e que foi atendido, uma vez que a biblioteca *Tensor Flow* tem uma versão oficial disponibilizada para *Java Script*.

3.1.3 BIBLIOTECAS

As bibliotecas utilizadas foram: "TensorFlow.js" para as funções de *machine learning*, "fft-js" para a geração das FFTs, "nodeplotlib" para a geração dos gráficos, "moving-averages" para a geração de médias móveis, "F_s" para a manipulação de arquivos externos com o *Node.js*, e "csv-append" para a inserção de dados de forma otimizada em um arquivo do tipo *Comma-separated Values* (CSV).

3.1.4 EDITOR DE CÓDIGO FONTE

Para a edição dos códigos, escolheu-se o *software* Visual Studio Code, que não é uma IDE (Ambiente de Desenvolvimento Integrado), e sim um editor de texto simples, porém, otimizado para programação de linguagens modernas, e com suporte a diversas ferramentas e *plugins*, como por exemplo a depuração de código, reconhecimento de palavras reservadas de muitas linguagens, integração com sistemas de versionamento de códigos, entre várias outras funções.

3.2 PADRONIZAÇÃO DOS DADOS

Um problema esperado para esta implementação é o efeito da falta da padronização dos dados, em que variações devido ao deslocamento no eixo do tempo, por

exemplo, podem fazer com que a rede neural considere dados que deveriam ser próximos, como diferentes, dificultando o processo de aprendizagem. Para que se possa minimizar este problema, os dados serão trabalhados no domínio da frequência, assim, duas coletas de um mesmo eletrodoméstico, por mais que tenham sido amostradas em instantes de tempos diferentes, terão o mesmo padrão no espectro de frequência.

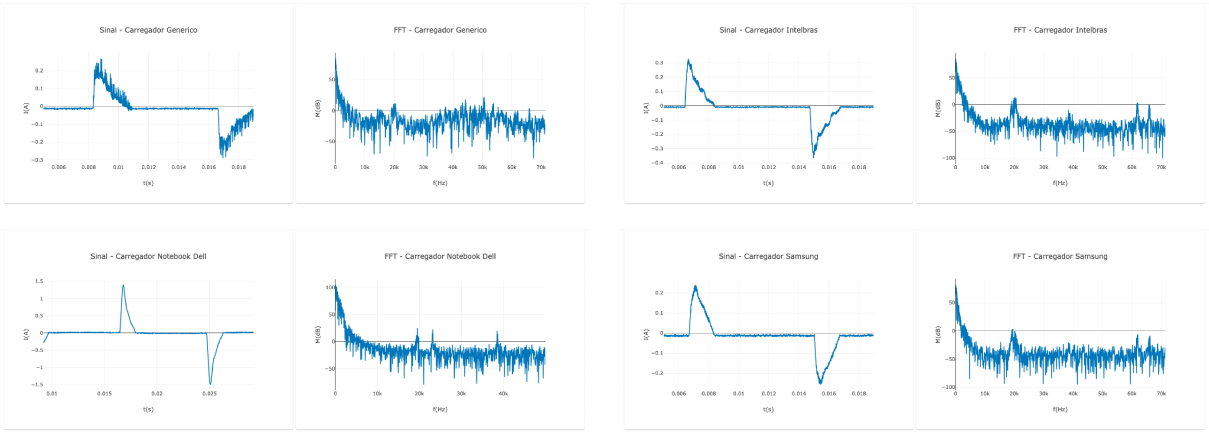
Para que o sinal seja convertido do domínio do tempo para o domínio da frequência, o mesmo será submetido a uma Transformada Discreta de *Fourier* (DFT), como uma etapa de pré-processamento da rede neural, e esta implementação será realizada em código.

O número de passos incrementais no eixo da frequência está diretamente relacionado à taxa de amostragem do sinal. Em outras palavras, quanto maior a taxa de amostragem, maior o número de frequências poderá ser observado no espectro gerado pelo pré-processamento. Este número carrega consigo a profundeza de informações sobre o sinal, e, para cada frequência, se tem um valor de magnitude associado. Essas duas informações serão armazenadas em um vetor unidimensional, uma de forma direta, e outra de forma indireta. A primeira será o valor da magnitude de determinada frequência, e será alocada dentro de uma posição do vetor. A segunda representa a frequência associada, e esta, será de forma indireta pois não haverá a necessidade de se guardar esta informação, visto que o passo entre uma frequência e outra é constante, logo, o próprio índice do vetor já nos dá a informação sobre qual frequência ele se refere. Neste sentido, um vetor completo conterá várias magnitudes associadas a várias frequências distintas. Cada vetor representa um eletrodoméstico.

3.3 TRATAMENTO DOS DADOS

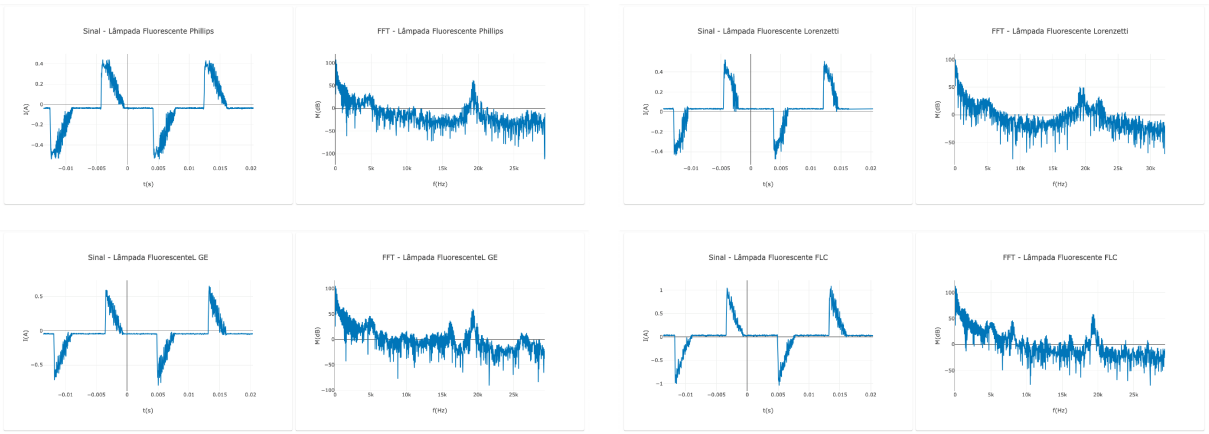
De posse das medições de corrente de todos os equipamentos, foi preciso padronizar os dados para que as entradas da rede neural tivessem o mesmo formato. Como o método escolhido para o treinamento foi o de que para cada sinal se tivessem vários neurônios na entrada onde cada um deles representa uma frequência do sinal, todos os dados precisam ter a mesma quantidade de frequências, e estas precisam estar igualmente espaçadas e com os mesmos valores no eixo das abscissas, para todos os equipamentos. Assim, o primeiro passo foi extrair o espectro de frequências de cada sinal. As Figuras 18, 19 e 20 mostram a obtenção do espectro de frequências de cada sinal por meio da implementação da transformada de *Fourier*.

Figura 18: Corrente e FFT para os 4 carregadores.



Fonte: O autor.

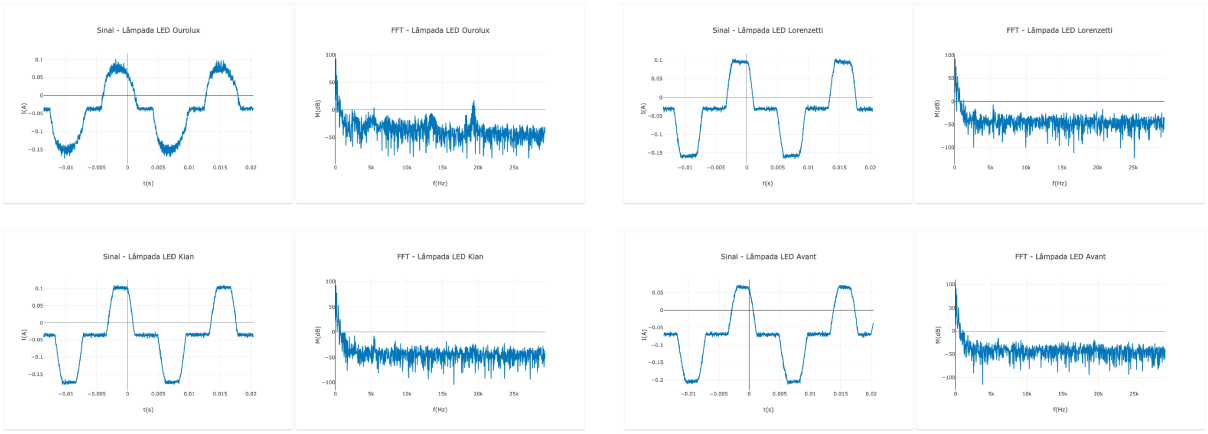
Figura 19: Corrente e FFT para as 4 lâmpadas fluorescentes.



Fonte: O autor.

Como pode ser visto nas Figuras 18, 19 e 20, o eixo de frequências dos equipamentos não são padronizados. Isto ocorre devido à taxas de amostragem diferentes, uma

Figura 20: Corrente e FFT para as 4 lâmpadas de LED.

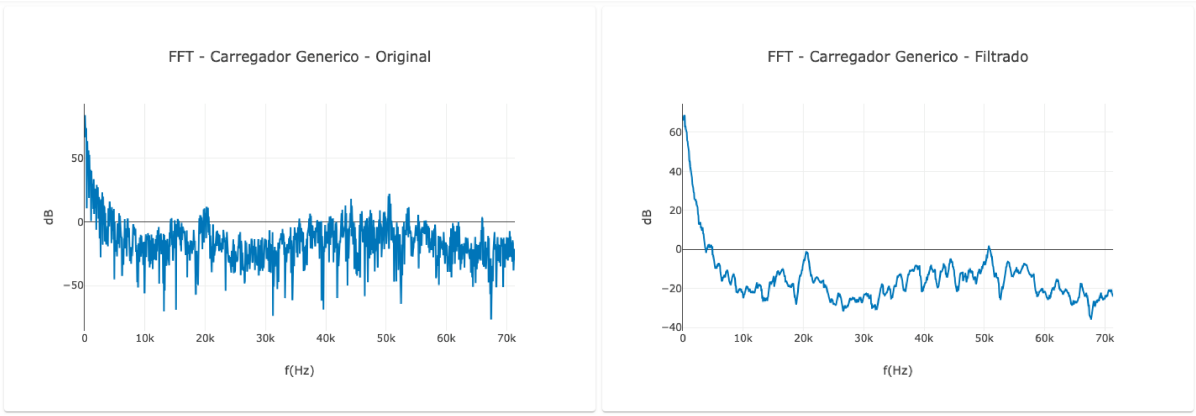


Fonte: O autor.

vez que o osciloscópio utilizado define uma frequência de amostragem automaticamente para cada medição, baseado na máxima frequência de oscilação que ele mesmo detecta quando se acopla um sinal à sua ponteira de leitura. Para que os eixos de frequência fiquem equalizados, alguns processos foram aplicados. Antes da normalização do eixo das abscissas, foi realizado um tratamento nas FFTs para eliminar altas componentes de frequência provenientes do processo de medição, componentes estas que não representam necessariamente as características intrínsecas do sinal, e sim espúrias do processo de medição. Como o objetivo é caracterizar o sinal com base em suas principais componentes, um sinal mais limpo, mas que ainda sim contenha diferenças significativas entre os aparelhos medidos, pode minimizar a "confusão" na etapa de classificação. O mecanismo usado para a filtragem foi o de média móvel, utilizando a função "dma" da biblioteca "moving-averages" do JS. Foi usado o parâmetro "alpha" de 0.1, ou seja, foram considerados 10 elementos na janela da média móvel (ZHANG, 2021). A filtragem foi aplicada em todas as FFTs, e o efeito pode ser visto no exemplo do carregador genérico, mostrado na Figura 21.

Para se resolver o problema da falta de equalização das frequências, escolheu-se o sinal com menor frequência de amostragem e utilizou-se de sua máxima frequência para definir as máximas frequências do eixo das abscissas dos demais. Os sinais com menor frequência foram encontrados nas duas categorias de lâmpada, onde o maior valor visto foi pouco maior do que 28KHz, sendo este o limite escolhido para o eixo padronizado das frequências. Como os sinais são discretos, cada valor de frequência gerado para cada sinal possui um valor de amplitude correspondente, não se sabendo assim a informação sobre o valor de amplitude de uma frequência vizinha não amostrada. Assim, no processo de equalização, definiu-se valores padrão para o eixo das frequências, então, realizou-se um processo de interpolação de primeiro grau entre frequências subsequentes para se estimar os novos valores dos eixos padronizados. A Figura 22 mostra o processo de como se dá a

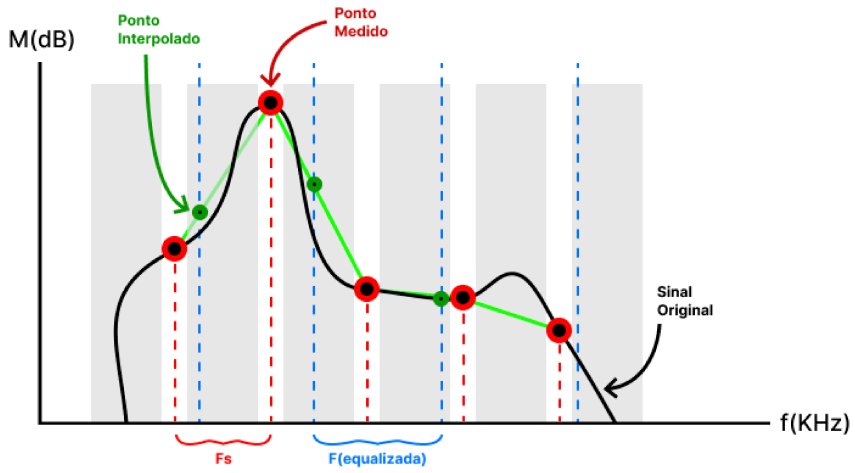
Figura 21: Comparação entre as FFTs do carregador genérico.



Fonte: O autor.

estimativa de um ponto que não pertence ao conjunto de dados amostrados, através da interpolação de primeira ordem. É montada uma equação do primeiro grau entre dois pontos amostrados, assim, pode-se estimar o valor da amplitude de qualquer frequência entre estes dois pontos. A Figura 22 é uma representação gráfica do que foi implementado em código dentro de uma função contendo um *loop*, para se poder estimar todas as novas frequências de todos os sinais.

Figura 22: Interpolação para equalização do eixo das frequências.



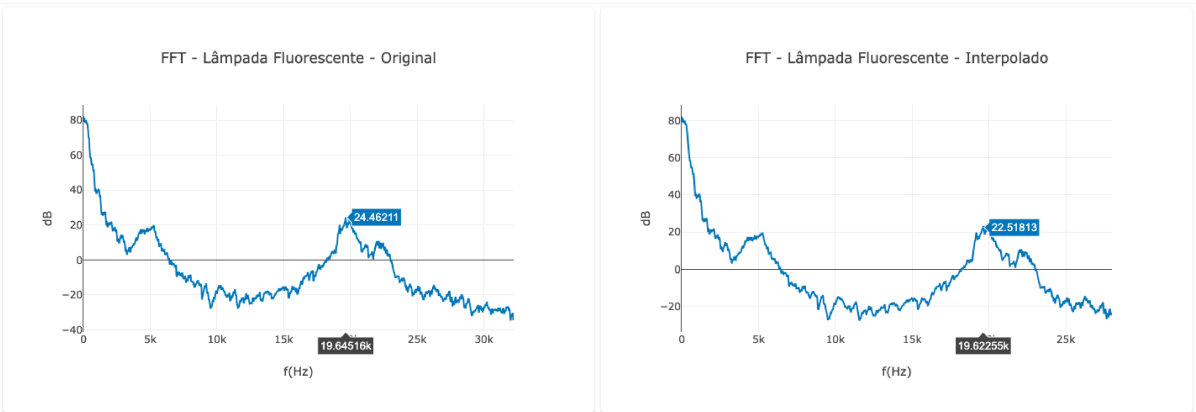
Fonte: O autor.

Para validar que a técnica de interpolação utilizada não traria grandes degradações nas principais características do sinal, plotou-se dois gráficos de um dos equipamentos, sendo um deles a FFT do sinal original (depois de passado pelo filtro de média móvel), e outro a FFT do novo sinal interpolado. Para se comparar se o sinal interpolado era fiel ao original, realizou-se uma interpolação onde as escalas máximas de frequências

eram próximas, sendo assim escolhida a lâmpada Fluorescente da marca *Lorenzetti*, que continha frequência máxima próxima de 28KHz. A comparação pode ser observada na Figura 23. Além da similaridade visual, que mostra que o sinal interpolado segue, em termos gerais, um comportamento próximo ao do original, um ponto específico foi utilizado para se comparar os dois dados. O ponto escolhido foi o que apresenta a maior amplitude (depois da de frequência zero), e este nos mostra que neste ponto, temos frequências próximas, e também valores de amplitudes próximos, como também visto na Figura 23. Uma vez que a técnica foi validada para o propósito deste projeto, aplicou-se a interpolação para todos os sinais, mesmo aqueles que possuíam frequências máximas próximas a 70KHz. Assim, todos os sinais tiveram seus eixos de frequência padronizados e limitados a 28KHz.

Outro parâmetro importante é o número de pontos escolhidos para o novo eixo. Se forem escolhidos mais pontos do que o sinal original contém, o sinal interpolado ficará mais "pesado", com mais informação, porém não existem perdas adicionais do que já é perdido na própria interpolação. No entanto, se utilizada uma quantidade menor de pontos do que existem no sinal original, quanto menor for a quantidade de pontos, maior será a perda. Como todos os sinais, independente da frequência de amostragem, obtinham 1000 pontos na FFT, o critério inicial para a escolha do número de pontos foi: utilizar a quantidade de pontos iguais à do sinal original. Assim, com 1000 pontos, os sinais que tinham frequência máxima próxima de 28KHz, tinham sua interpolação com aproximadamente a mesma quantidade do sinal original. Já os sinais que possuíam frequências máximas próximas de 70KHz por exemplo, tiveram parte do espectro descartado, e entre uma frequência original e outra, vários pontos foram utilizados na interpolação deste intervalo. Novas interpolações com diferentes quantidades de pontos foram testadas posteriormente para verificar se os resultados de classificação da rede neural performariam melhor.

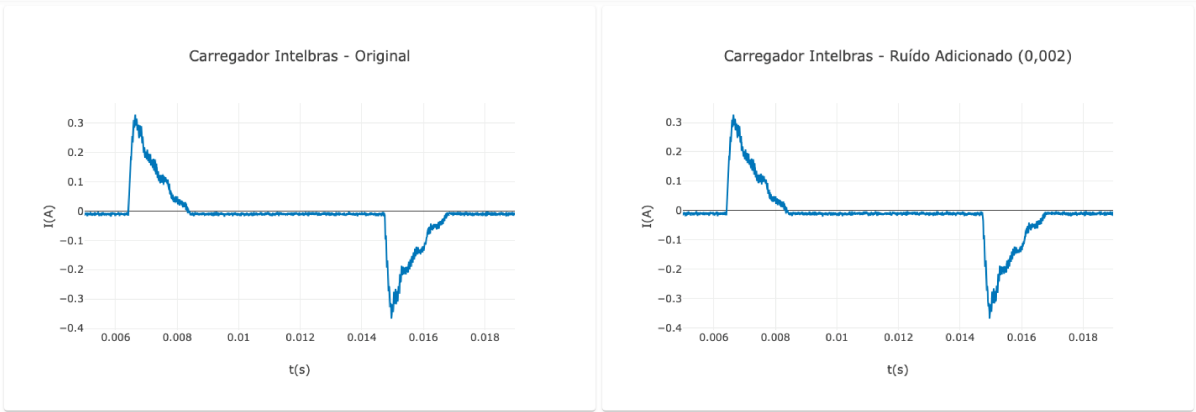
Figura 23: Comparação entre sinal original e sinal interpolado.



Fonte: O autor.

De posse dos sinais equalizados com relação ao eixo das frequências, o próximo passo foi a preparação dos dados para o treinamento da rede neural. O número de dados é pequeno (12 equipamentos) e quando se deseja realizar a predição de um equipamento que não tenha sido utilizado previamente nos dados de treinamento, tem-se um equipamento a menos de cada grupo. Desta forma, foi realizada uma técnica para aumentar o número de dados, chamada de *data augmentation*, que tem o intuito de se criar réplicas do sinal, porém, com pequenas variações entre eles, de forma a se ter sinais diferentes com características gerais iguais. Para tal implementação foi criada uma função responsável por adicionar um ruído no sinal, inserindo valores aleatórios ao longo de toda a amostra, gerando pequenas variações ao longo de todas as frequências do espectro. A função criada para tal permitia que a intensidade do ruído fosse controlada por uma constante multiplicativa, assim, foi possível testar o efeito de diferentes níveis de ruído, e o efeito destes níveis no comportamento da rede neural em seu treinamento e predição. As Figuras 24 e 25 mostram a comparação do sinal original com diferentes níveis de ruído adicionados.

Figura 24: Comparação entre sinal original e sinal com menor ruído adicionado.

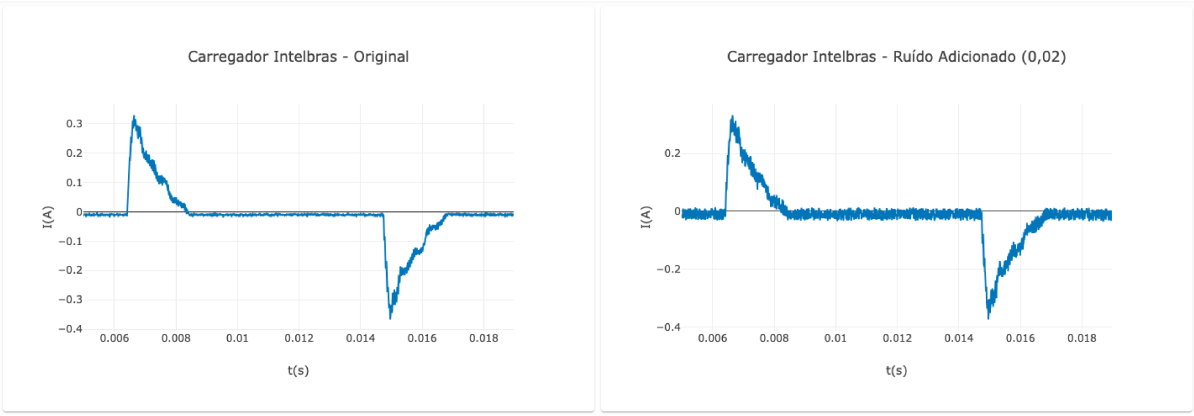


Fonte: O autor.

O número de réplicas dos sinais originais é um fator que posteriormente foi testado, para que se pudesse analisar se esta quantidade faria diferença no treinamento/predição da rede neural. Nesta etapa, como configuração inicial, estabeleceu-se arbitrariamente o número de 4 réplicas para cada sinal. As Figuras 26 e 27 nos mostram o efeito destes ruídos para um equipamento, onde cada cor representa uma réplica. Pode-se observar que quanto maior o nível de ruído, maior é o desvio padrão entre as FFTs, mas ainda sim, os sinais apresentam características gerais similares.

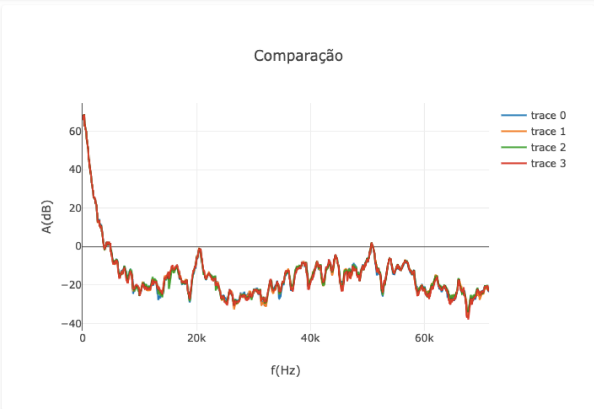
O procedimento de replicação foi realizado para todos os sinais, onde assim, inicialmente se teriam 48 amostras padronizadas para compor o grupo de treinamento.

Figura 25: Comparação entre sinal original e sinal com maior ruído adicionado.



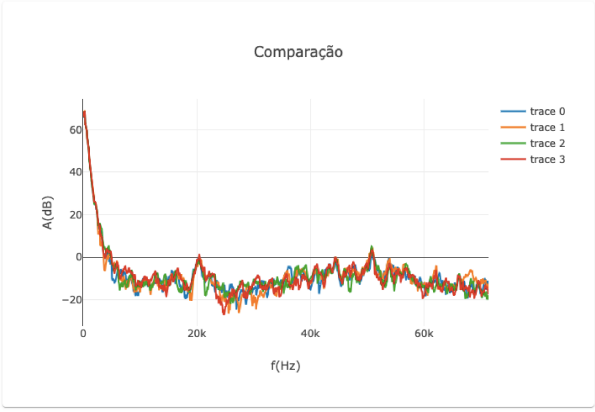
Fonte: O autor.

Figura 26: FFTs de 4 réplicas ruidosas com menor ruído adicionado.



Fonte: O autor.

Figura 27: FFTs de 4 réplicas ruidosas com maior ruído adicionado.



Fonte: O autor.

3.4 IMPLEMENTAÇÃO DA REDE NEURAL

Para esta aplicação, a rede neural foi construída de forma a ter muitos neurônios na camada de entrada, onde cada um deles está associado a uma frequência da FFT, que foi obtida e armazenada, como explicado na seção anterior. Foram testadas algumas variações de implementação para se comparar e descobrir qual a forma em que a rede neural entrega um melhor resultado. Variações como por exemplo: comprimento das amostras, escolha do nível de ruído, função de ativação de cada camada, quantidade de camadas ocultas, quantidade de épocas, entre outros. Estas variações iniciais foram parte de testes que tinham o objetivo de definir um *setup* que se seguiria inalterado nas próximas etapas. O objetivo de tais testes foram de descobrir se era melhor para um resultado final de classificação, usar ou não a média móvel, descobrir qual a quantidade de réplicas ruidosas se precisava usar, e também a partir de qual quantidade já se passava a não fazer mais diferença. Descobrir qual o nível do ruído para se adicionar nas réplicas, e também o número de épocas que a rede neural iria executar. O motivo pelo qual se desejou encontrar uma configuração inicial inalterada, foi o de que as próximas etapas fossem focadas somente na configuração da arquitetura da rede neural, como por exemplo a estrutura da rede, se é convolucional ou não, as funções de ativação de entrada, entre outros parâmetros. Para estes primeiros testes, chegou-se em uma configuração otimizada, e o *setup* pode ser visto na Figura 28.

Figura 28: Configuração de todos os testes a serem realizados.

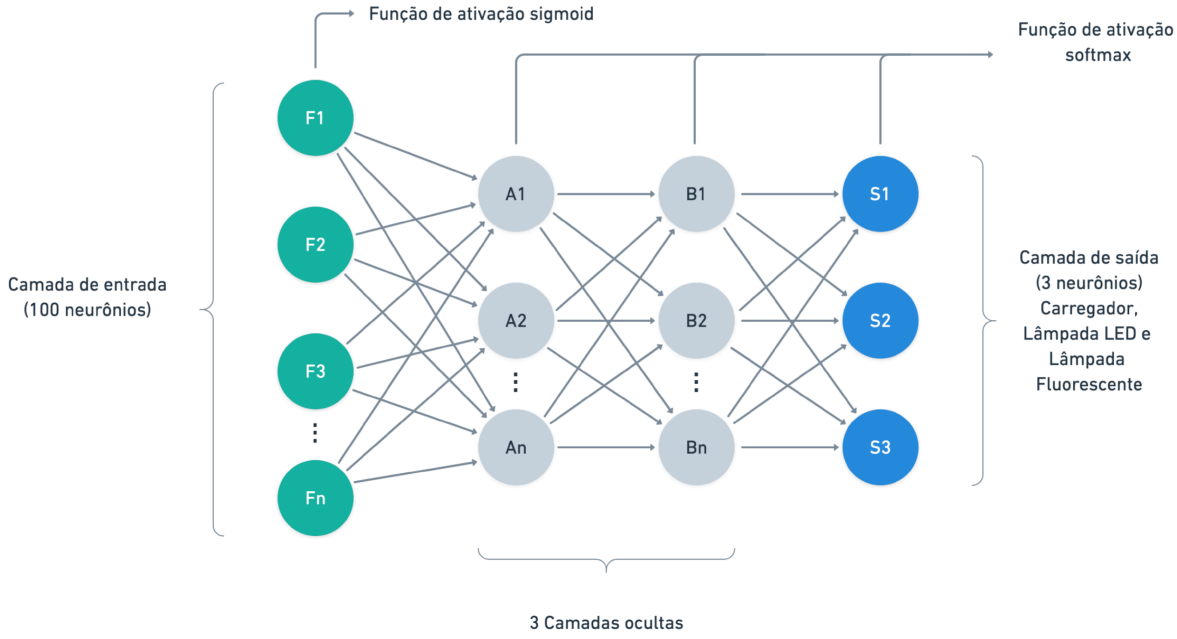
Configurações dos Testes	
Média Móvel na FFT	Sim
Nível do Ruído	0,02
Quantidade de Réplicas Ruidosas	10
Tamanho do Vetor de Cada Amostra	100
Épocas	10
Testes para Cada Amostra	10

Fonte: O autor.

Após a primeira configuração, o próximo passo é a alternância entre as arquiteturas das redes. Viu-se que todas as redes performaram melhor com relação à quantidade de acertos quando tinham uma função de ativação *softmax* nas camadas de saída. Esta função de ativação é recomendada para casos onde se deseja classificar elementos com relação a alguma probabilidade. Quando comparada com outras funções de ativação nas camadas de saída, independente da arquitetura da rede, a função *softmax* trouxe melhores taxas de acerto em todos os casos, sendo assim esta também uma configuração inalterada daqui por diante. Outra variação importante foi a do tipo de função de perda utilizada. Testou-se dois tipos, a de entropia cruzada, e a de erro médio quadrático, e os resultados sempre foram mais promissores utilizando-se a do primeiro tipo. Assim, foram testados três diferentes combinações práticas: uma rede não convolucional, e duas redes

convolucionais com diferentes funções de ativação na camada de entrada: uma com uma função *ReLU*, que geralmente é utilizada em conjunto com a função *softmax* na camada de saída para classificação de mais de duas categorias, e outra com a função *sigmoid*, que também é uma utilização comum em redes neurais de classificação. Todos os testes para estes três casos foram feitos em duas situações: uma em que o equipamento a ser predito participava do grupo de treinamento, e outra em que o mesmo não participava. Este último é considerado o caso mais severo e também mais valioso, pois se a rede neural for capaz de classificar corretamente um equipamento que não fez parte do grupo de treinamento, pode-se considerar que ela tem uma boa eficácia. As arquiteturas das redes testadas podem ser observadas nas Figuras 29, 30 e 31.

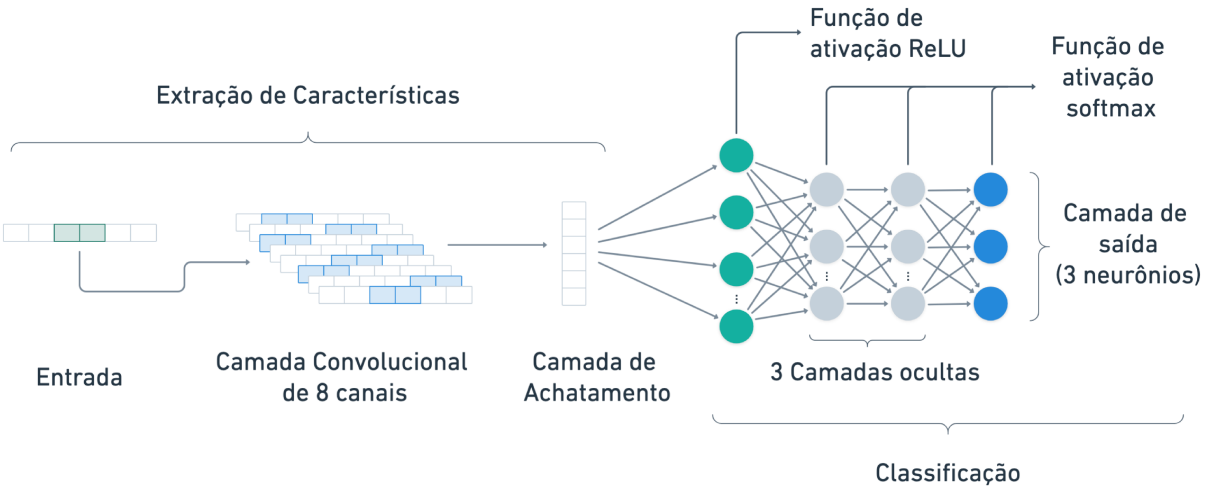
Figura 29: Rede Neural não convolucional.



Fonte: O autor.

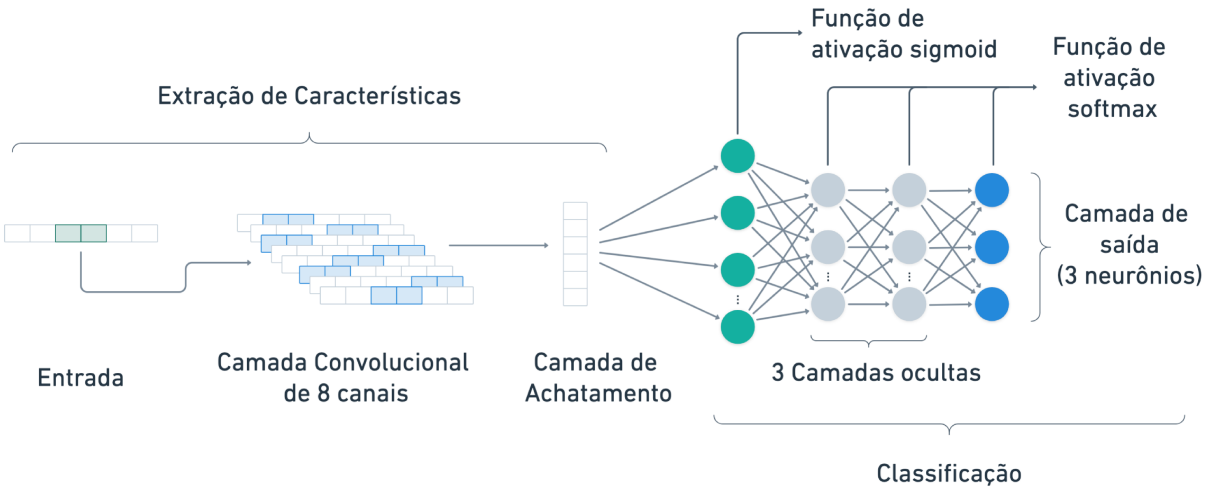
Outro ponto importante foi o processo utilizado para o teste e coleta de resultados. Foram implementadas variações dos métodos *full-training* e *leaving-out* para organizar o treinamento, em que, realizaram-se através de um laço de repetição, 10 iterações de treinamento e predição para cada um dos doze equipamentos (além das épocas já realizadas por cada treinamento). O objetivo era ter uma quantidade maior de resultados para que as estatísticas se tornassem mais confiáveis. Testou-se também variações de 30 e de 100 iterações, porém, viu-se que o resultado era muito próximo do realizado com 10, e, como o custo computacional é elevado e os testes com mais de 10 repetições demoravam muito para serem executados, 10 foi o número ótimo escolhido para a quantidade de vezes que cada equipamento fosse submetido a um ciclo de treinamento e predição.

Figura 30: Rede Neural convolucional com ReLU.



Fonte: O autor.

Figura 31: Rede Neural convolucional com Sigmoid.



Fonte: O autor.

Para o grupo de treinamento, utilizaram-se apenas as FFTs das réplicas ruidosas de cada um, e para a predição, utilizou-se a FFT original de cada sinal. Quando o elemento a ser predito participava do treinamento, sua réplica ruidosa era quem fazia parte do grupo de treinamento, assim foram utilizadas 120 amostras para treinamento, e 1 amostra (original) para predição. Quando o elemento não participava do grupo de treinamento, na prática, eram suas réplicas ruidosas que não estavam participando, assim se tinham 110 elementos no treinamento, e 1 (original) na predição.

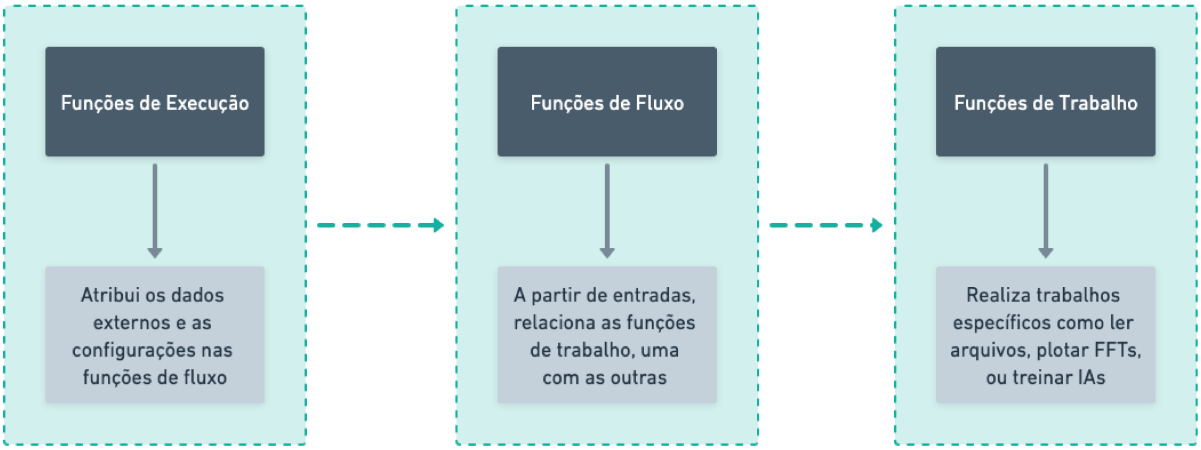
As redes neurais, independente da arquitetura, funções de perda, *backpropa-*

gation, entre outros, foram todas implementadas utilizando-se as funções da biblioteca *Tensorflow.js*, para *Node.js*, escritas em *Java Script*. Estas funções possuem uma grande abstração, não sendo visível o que está sendo realizado "por debaixo dos panos". O uso consiste em se passar parâmetros de entrada para as funções e analisar os resultados de saída, o que por um lado é bom pois facilita e agiliza o processo de implementação, porém, por outro lado dificulta um pouco o entendimento. Contudo, a documentação da biblioteca fornecida pelo site oficial é bem completa e explicativa, onde são mostradas as dinâmicas internas de cada função e como são implementadas, o que faz um perfeito casamento com a teoria vista neste trabalho e explicada na seção de fundamentação teórica. Além disso, os exemplos práticos contidos no site oficial fornecido pelo *Google* também oferecem um bom direcionamento sobre a implementação que deverá ser feita. Alguns destes exemplos ajudaram a guiar as primeiras linhas de código desenvolvidas neste trabalho.

3.5 ARQUITETURA DO CÓDIGO

Para o desenvolvimento dos códigos deste trabalho, utilizou-se uma arquitetura de camadas, separadas por uma dinâmica semelhante ao que acontece no conceito de orientação a objetos, em que as funções que executam tarefas importantes ficam em arquivos separados, e organizadas por "classes", ou, neste caso, por "tipos de trabalho". A Figura 32 mostra a separação entre as camadas implementadas.

Figura 32: Arquitetura do código.

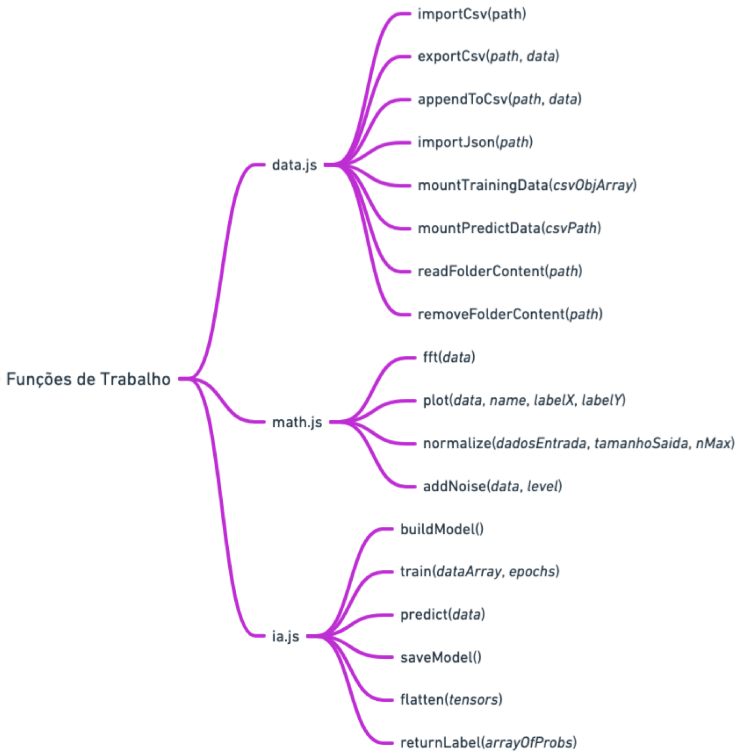


Fonte: O autor.

O intuito deste tipo de arquitetura é que haja um desacoplamento entre as funções. Por exemplo, para uma função que realiza o trabalho de tratar os dados de um CSV, não é interessante que a mesma tenha relação com uma função que realiza uma operação matemática como uma FFT. Além disso, é interessante que o uso de uma função não dependa de sua implementação interna. Assim, caso se precise alterar a forma como

uma função realiza determinada tarefa, o escopo principal da aplicação dificilmente será prejudicado. Este caso citado ocorreu algumas vezes neste trabalho, onde se precisou mudar o tipo de rede neural utilizada, ou mesmo a forma como os gráficos eram gerados, entre outros. Como pode ser visto na Figura 33, a primeira camada de implementação, também chamada de "Funções de Trabalho", é uma camada de mais "baixo nível", onde se realizam detalhes técnicos, como implementações matemáticas, tratamentos de dados e funções de IA. No arquivo "math.js" por exemplo, existem funções para se converter um vetor de dados em um vetor de FFT e retornar o resultado. Existem também funções para plotar os dados em um gráfico, normalizar um vetor através de interpolação, entre outros.

Figura 33: Árvore das funções de trabalho, separadas em arquivos JS.



Fonte: O autor.

O escopo geral do arquivo "math.js" pode ser visto no Código 1. Cada uma destas funções realizam algumas tarefas por meio de outras bibliotecas comumente utilizadas pela comunidade do *Node.js*, como pode ser visto nas linhas de "require", no topo do arquivo mostrado no Código 1. Da mesma forma que foi mostrado para as funções matemáticas, tem-se também o código "data.js", que é responsável por realizar trabalhos com arquivos externos, como por exemplo importar um CSV e retornar seus dados na forma de um objeto *Java Script*, ou então exportar uma informação que está disponível a nível de execução de código (por exemplo um objeto *Java Script* atribuído a uma variável

que está na memória RAM) para um arquivo CSV externo. Outras funções pertinentes para a realização do projeto como por exemplo a leitura do conteúdo de uma pasta, e similares, também estão presentes em "data.js", como pode ser visto na Figura 33. No código "ia.js" tem-se as funções responsáveis por definir a arquitetura do modelo de rede neural, realizar o treinamento e realizar uma predição por meio de um dado de entrada. Os tipos de elementos a serem treinados, ou preditos, não fazem diferença nesta camada de implementação, assim, caso no futuro se deseje utilizar este mesmo código para trabalhar com outro tipo de informação, com por exemplo amostras de áudio, ao invés de amostras de corrente elétrica, se poderia fazê-lo, bastando apenas que se configure alguns parâmetros nas camadas posteriores, que são as responsáveis pelas funções de fluxo, e pela execução geral da aplicação.

Código 1: Escopo geral do arquivo "math.js"

```
1  const plot = require('nodeplotlib');
2  const FFT = require('fft-js');
3  const ma = require('moving-averages');
4
5  module.exports = {
6
7      fft(data){ ...
8      },
9
10     plot(data, name, labelX, labelY){ ...
11     },
12
13     normalize(dadosEntrada, tamanhoSaida, nMax){ ...
14     },
15
16     addNoise(data, level){ ...
17     }
18
19 };
```

As camadas posteriores são as de fluxo e de execução. Na camada de funções de fluxo foram definidas as tarefas que relacionam as funções da camada de trabalho, mas que ainda não lidam com nenhum tipo de dado externo. São funções que foram preparadas para receber um dado de entrada, executar uma sequência de operações das camadas de trabalho e retornar um valor de saída, como pode ser visto na Figura 34. Um exemplo de função desta camada, é a função "trainAndPredict", que recebe como parâmetro de entrada o caminho de um diretório que contenha as amostras de treinamento, o caminho do arquivo a ser predito, o número de épocas a serem executadas no treinamento, e um objeto JS contendo os caminhos dos elementos que devem ser excluídos do grupo de treinamento (pode-se passar um objeto vazio caso não se queira excluir nenhum). Esta

função é responsável por ler os arquivos de treinamento presentes em uma pasta, realizar a montagem do modelo de arquitetura da rede neural, realizar o treinamento dos dados obtidos da pasta, realizar a predição de um item, e, por fim, retornar a etiqueta do elemento que foi predito. Sua implementação interna pode ser vista no Código 2.

Código 2: Função "trainAndPredict".

```
async trainAndPredict(trainingDataFolder, fftCsvToPredictPath, epochs,
    excludeTrainItems){

     const aPromise = new Promise(async (resolve, reject) => {

         const trainingData = await this.returnLabeledDataByFoder(
            trainingDataFolder, excludeTrainItems);

         await this.buildModel().then((log) => {

             console.log(log);

             this.train(trainingData, epochs).then(() => {
                 this.predict(fftCsvToPredictPath).then(async (prediction
                    ) => {
                     const predictionLabel = prediction;
                     resolve(predictionLabel);
                 });
             });

         });

     });

     return aPromise;

}
```

Pode-se observar no Código 2 uma estrutura de código encadeada que trabalha com o conceito de chamadas assíncronas no *Java Script*, recurso este que é muito utilizado ao longo de todos os códigos. Este tipo de implementação é utilizado quando se tem alguma tarefa que não tem um retorno imediato, como é o exemplo da leitura de um arquivo no disco por exemplo, ou então do treinamento de uma rede neural, que dependendo da configuração, pode demorar alguns minutos. Uma função assíncrona pode ser utilizada de várias formas, e na função mostrada no Código 2, podemos ver duas delas, uma onde se usa o comando "await" para fazer com que o restante do código "espere" até que aquela linha finalize sua execução, e também o comando ".then()", que é utilizado acoplado a uma função assíncrona para que determinada ação seja realizada dentro dele, logo que a execução da função externa finalize. Na prática, o código a ser execu-

Figura 34: Árvore das funções de fluxo.

Fonte: O autor.

tado posteriormente é chamado dentro dos parenteses do ".then()", onde se deve escrever uma nova função. A escrita originalmente é definida por ".then(function(){ trecho a ser executado })", porém, existe uma simplificação comumente utilizada que faz o mesmo papel, definida por ".then(() => { trecho a ser executado })". Esta simplificação é chamada de "arrow function". Nesta "arrow function" pode-se passar, no primeiro parenteses, parâmetros que são retornados pela função externa em que se acoplou o ".then()". Por exemplo, se a primeira função é responsável por ler os dados de um arquivo CSV, e esta retorna um objeto JS contendo os dados lidos, pode-se chamar no primeiro parênteses da "arrow function" este resultado, atribuindo qualquer nome a uma variável que pode ser usada dentro da nova função interna, estando assim disponível para se trabalhar com ela.

Geralmente se usa o termo “result”, para este tipo de variável. Na prática, funcionaria como: “importCsv(’/caminho/do/arquivo’).then((result) => { print(result); })”.

Outra implementação muito utilizada, que também pode ser vista no Código 2, é o “new Promise()”. Esta é uma estrutura que tem o objetivo de retornar uma resposta de uma função assíncrona. Quando se atribui a uma variável o resultado de uma função do tipo “Promise”, se está definindo a promessa de um retorno. Assim, se quisermos exibir o valor contido em uma variável em que se atribuiu uma “Promise”, devem-se utilizar as estruturas de “await”, ou então de “.then()”. Os recursos de execuções de tarefas assíncronas do *Java Script* são muito importantes para se trabalhar com funções que não podem ser processadas em tempo real, e foram muito úteis para este trabalho.

As funções escritas em “flowfunctions.js” são executadas pela última camada, que é a de execução, e nesta, são atribuídos os dados externos nas funções de fluxo. A Figura 35 mostra alguns exemplos de funções presentes nesta camada, que estão inseridas no arquivo “app.js”. Além disso, alguns objetos JS foram definidos para as configurações desejadas para execução, como pode ser visto no Código 3, onde foram escritos os parâmetros de teste citados nas seções anteriores. Os caminhos das pastas contendo os dados de treinamento, e os dados originais também estão presentes nestes objetos. Um exemplo de implementação desta camada é a função “predictAll”, representada no Código 4. Esta função, tem como objetivo ler os parâmetros do objeto de configurações globais, definir o item a ser predito, realizar um laço *for* que roda os testes pelo número de vezes definido no objeto de configurações, chamar a função de treinamento e predição passando os dados externos que foram lidos, e exportar os resultados para arquivos CSV externos de análise, além de mostrar os dados na tela, como mostrado no Código 5.

Código 3: Configurações globais definidas em um objeto JS.

```
1  const globalParams = {
2
3      "noiseLevel" : 0.02,
4      "generatedLength" : 100,
5      "nGenerateTraining" : 10,
6      "nGeneratePredict" : 10,
7      "epochs" : 10,
8      "tests" : 10,
9      "inputFolderPrefix": "./data/input/cleared",
10     "outputFolderPrefix": "./data/output",
11     "trainingDataFolder": "./data/output/generated/training",
12     "predictDataFolder": "./data/output/generated/predict",
13     "items" : {
14         "1" : {item : "CarregadorGenerico", label : 0},
15         "2" : {item : "CarregadorIntelbras", label : 0},
16         "3" : {item : "CarregadorNotebookDell", label : 0},
17         "4" : {item : "CarregadorSamsung", label : 0},
18
```

Figura 35: Árvore das funções de execução.

Fonte: O autor.

```
        "5" : {item : "LED_Ourolux", label : 1},
        "6" : {item : "LED_Lorenzetti", label : 1},
        "7" : {item : "LED_Kian", label : 1},
        "8" : {item : "LED_Avant", label : 1},

        "9" : {item : "FL_Phillips", label : 2},
        "10" : {item : "FL_Lorenzetti", label : 2},
        "11" : {item : "FL_GE", label : 2},
        "12" : {item : "FL_FLC", label : 2},
    }
};
```

Código 4: Função “predictAll”.

```
async function predictAll(iterations, mode){

    const localResults = [];
    const globalReults = [];

    var excludeTrainingItemsObject = {};
    var item;
    var label;
    var success = false;

```

```
11      localResults.push('Configs: ' + JSON.stringify(globalParams));
12      globalReults.push('Configs: ' + JSON.stringify(globalParams));
13
14      for(var i = 0; i < Object.values(globalParams.items).length; i++){
15
16          var itemSuccessRate = 0;
17
18          item = Object.values(globalParams.items)[i].item;
19          label = Object.values(globalParams.items)[i].label;
20
21          if(mode == "exclude"){
22              excludeTrainingItemsObject = {
23                  "1" : item
24              }
25          }
26
27          const predictItem = './data/output/fft/' + item + '.csv';
28
29          for(var j = 0; j<iterations; j++){
30
31              await flow.trainAndPredict(globalParams.trainingDataFolder,
                    predictItem, globalParams.epochs,
                    excludeTrainingItemsObject).then((response) => {
32
33                  if(label == response){
34                      success = 'sim';
35                      itemSuccessRate += 100;
36                  }else{
37                      success = 'não';
38                  }
39
40                  localResults.push('Acertou: ' + success + '. Entrada ('
                        + mode + '): [' + predictItem + '] (' + label + ') ->
                         ' + '(' + response + ') ' + flow.mapLabelToName(
                        response));
41
42              });
43
44          }
45
46          globalReults.push('Taxa de acerto (' + mode + ') - ' + item + ':
                ' + itemSuccessRate/iterations + '%');
47
48      }
49
50      await flow.appendAnalyticsToCsv({
51          x : new Date().toLocaleString(),
```

```
52          y : JSON.stringify(globalReults)
53      }, './data/output/analytics/allAnalytics.csv');
54
55      await flow.appendAnalyticsToCsv({
56          x : new Date().toLocaleString(),
57          y : JSON.stringify(localResults)
58      }, './data/output/analytics/a0.csv');
59
60      await flow.saveModel();
61
62      console.log(globalReults);
63
64  }
65
66  predictAll(globalParams.tests, "exclude");
```

Código 5: Resultado da execução da função “predictAll”.

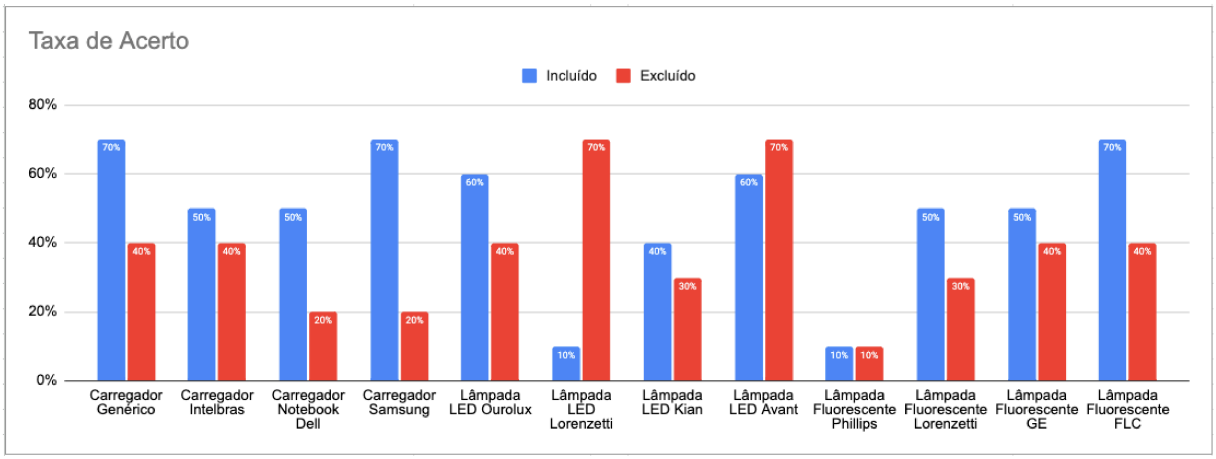
```
1   'Taxa de acerto (exclude) - CarregadorGenerico: 100%',
2   'Taxa de acerto (exclude) - CarregadorIntelbras: 100%',
3   'Taxa de acerto (exclude) - CarregadorNotebookDell: 100%',
4   'Taxa de acerto (exclude) - CarregadorSamsung: 100%',
5   'Taxa de acerto (exclude) - LED_Ourolux: 100%',
6   'Taxa de acerto (exclude) - LED_Lorenzetti: 100%',
7   'Taxa de acerto (exclude) - LED_Kian: 100%',
8   'Taxa de acerto (exclude) - LED_Avant: 100%',
9   'Taxa de acerto (exclude) - FL_Phillips: 0%',
10  'Taxa de acerto (exclude) - FL_Lorenzetti: 100%',
11  'Taxa de acerto (exclude) - FL_GE: 100%',
12  'Taxa de acerto (exclude) - FL_FLC: 100%'
```

Outras funções não citadas estão presentes no arquivo “app.js”, e serviram de apoio para a realização deste trabalho, no sentido de plotar gráficos, gerar as réplicas ruidosas e armazenar em pastas, gerar as FFTs dos arquivos originais, entre várias outras aplicações que podem ser vistas na Figura 35. As funções presentes em “app.js” foram utilizadas uma por vez. Para cada etapa do trabalho realizavam-se determinadas execuções, e extraíam-se os resultados, assim, o código possui muitas funções que não estão sendo chamadas, porém, em algum momento elas foram utilizadas. Por meio do encadeamento das camadas citadas acima, e da inserção dos dados coletados para os 12 aparelhos, chegou-se no objetivo deste trabalho, que era predizer o tipo de um eletrodoméstico com base na sua corrente elétrica. Os resultados obtidos são mostrados e discutidos na próxima seção.

4 RESULTADOS

Após a implementação e execução dos códigos citados acima, obtiveram-se diferentes resultados para os diferentes tipos de redes neurais utilizadas. Para todos os testes realizados, as taxas de acerto foram sempre superiores quando o equipamento a ser predito participava do grupo de treinamento. Para a rede neural não convolucional, as taxas de acerto para todos os equipamentos, tanto nas análises que incluíam o equipamento a ser predito no grupo de treinamento, quanto nas que excluíam, podem ser vistas na Figura 36. Observa-se que para a maior parte dos equipamentos, a taxa de acerto no caso de incluído foi próxima a 50%, o que demonstra uma razoável eficácia deste modelo, visto que o problema tem 3 classes. Para o caso de excluído, este tipo de rede não se mostrou eficiente, apresentando uma taxa de acerto próxima de 37%. Pode-se concluir que, neste caso, este tipo de rede poderia ser útil para onde se tem uma base de dados muito extensa, e que, supostamente, já contenha todos os elementos que se deseja analisar. Caso não houvesse outras alternativas de modelos de rede neural, este cenário pode ser uma realidade futura. Porém, não se mostra viável para um início de implementação do projeto.

Figura 36: Taxa de acerto da rede não convolucional.

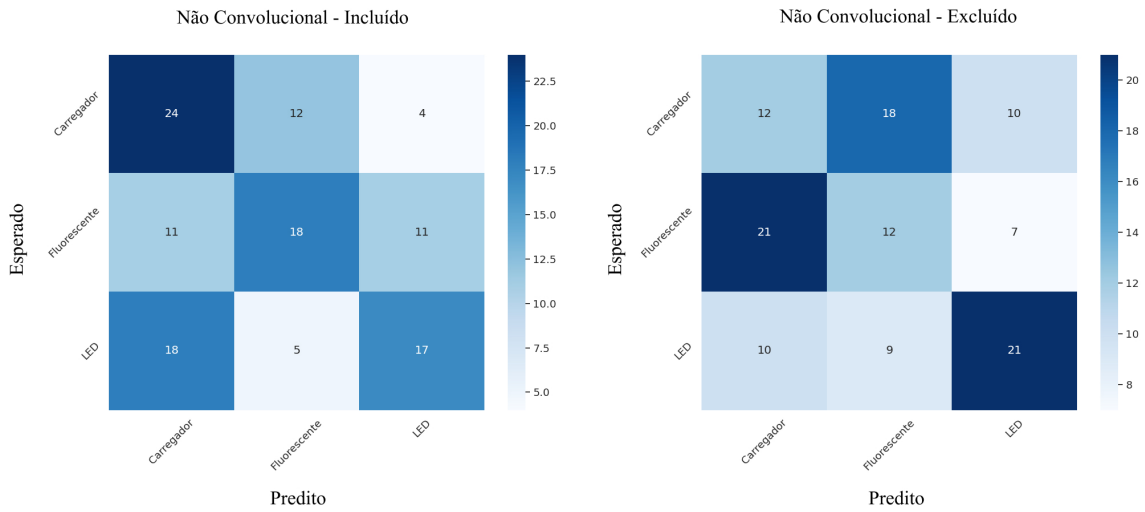


Fonte: O autor.

Além da taxa de acerto para cada equipamento, foram geradas as matrizes de confusão para cada tipo de rede. A matriz de confusão nos fornece uma compreensão sobre como a rede está se comportando no processo de predição, exibindo a informação sobre a quantidade de acertos, e também com quais equipamentos os erros foram confundidos. Esta relação é feita com base nos elementos das linhas e das colunas da matriz, onde em cada linha se tem uma entrada, representado uma resposta esperada, e em cada coluna se tem um resultado obtido. A diagonal principal contém os acertos, uma vez que o elemento esperado é igual ao resultado. As demais posições representam a quantidade de vezes em

que se obteve uma saída diferente da esperada, e também qual foi o elemento confundido. Assim, para o caso dos erros, pode-se ver não somente o erro, mas também em quais pares se teve uma maior confusão. Em um caso perfeito, onde todas as predições estão corretas, a matriz teria a diagonal principal com todas as contagens máximas, e os demais valores zerados. Quanto mais dispersos forem os números, menos eficiente é a rede. A Figura 37 mostra as matrizes de confusão da rede não convolucional para os casos de incluído e excluído. Observa-se que para o caso de incluído, a matriz ainda contém a maior parte dos valores na diagonal principal, porém, houve muitas ocorrências de confusão. No caso de excluído, o resultado foi ainda pior, mostrando muita confusão entre todos os elementos.

Figura 37: Matrizes de confusão da rede não convolucional.



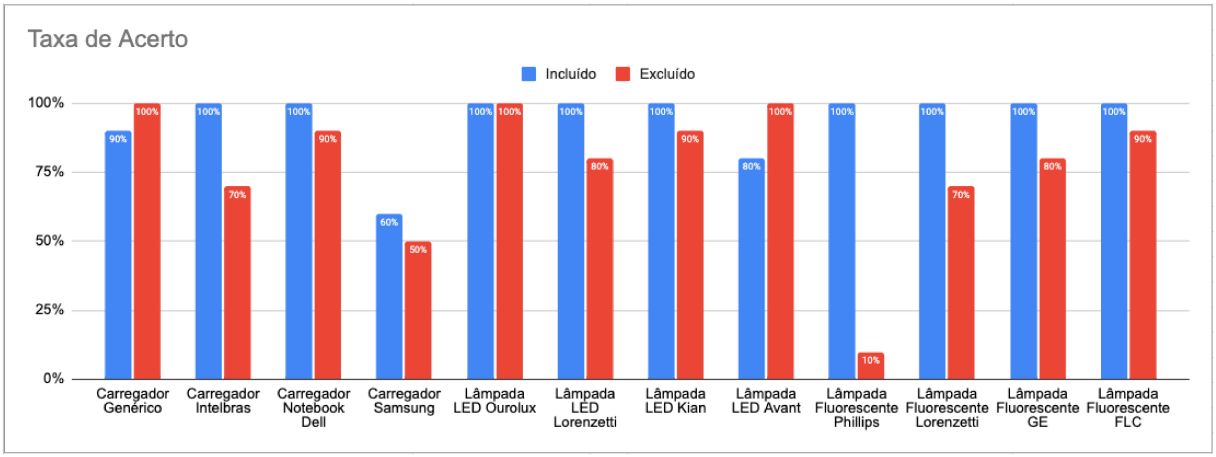
Fonte: O autor.

Para a implementação da rede convolucional com uma função de ativação *ReLU* nos neurônios da camada de entrada, os resultados podem ser vistos na Figura 38. Observa-se um aumento expressivo na taxa de acerto geral, quando comparado com a rede não convolucional. Neste caso, tanto para os testes com inclusão quanto para os com exclusão, a maior parte dos equipamentos tiveram sua taxa de acerto acima de 50%. Quando analisado somente o caso de quando incluso, quase todos os resultados foram próximos de 100%, o que mostra uma ótima eficácia deste tipo de rede.

As matrizes de confusão desta rede podem ser vistas na Figura 39, onde observa-se uma quantidade menor de confusão entre os elementos, sendo o melhor resultado para o caso de incluído, e certa taxa de confusão para o caso de excluído, mas ainda sim, de forma mais eficiente do que a rede não convolucional.

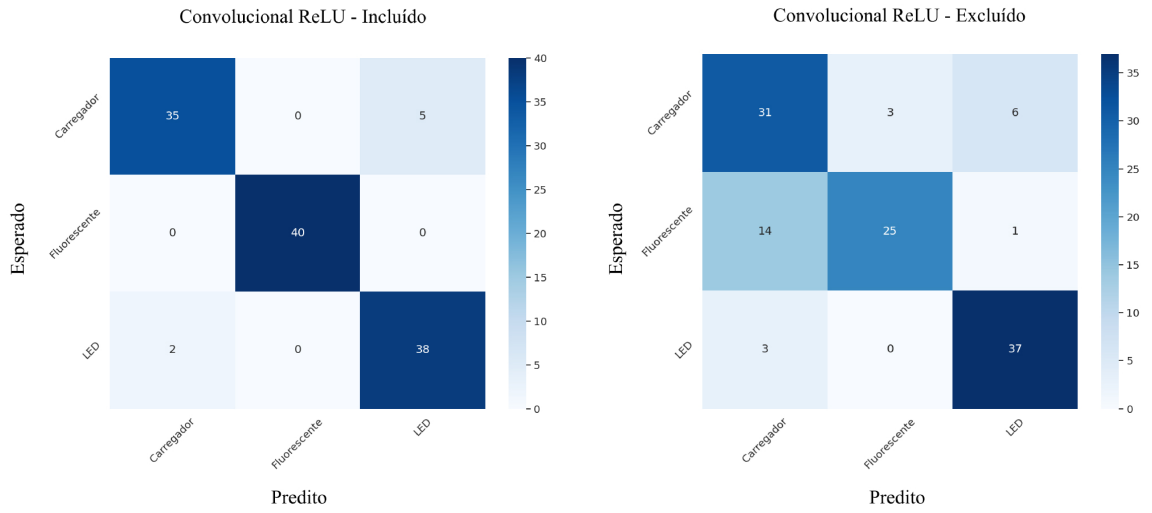
Para a implementação da rede convolucional com função de ativação *Sigmoid* nos neurônios da camada de entrada, os resultados podem ser observados na Figura 40. Esta rede mostrou os melhores resultados de todos os testes. Pode-se observar uma taxa de acerto de 100% em todos os eletrodomésticos para o caso de incluído, e uma taxa de

Figura 38: Taxa de acerto da rede convolucional - *ReLU*.



Fonte: O autor.

Figura 39: Matrizes de confusão da rede convolucional - *ReLU*.



Fonte: O autor.

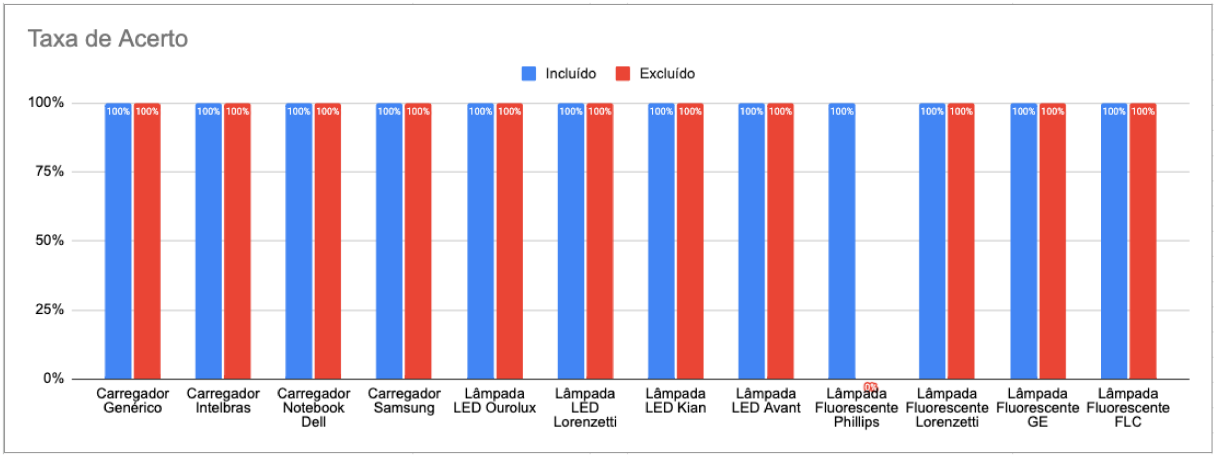
acerto de 100% em onze dos doze equipamentos. Houve apenas um dos equipamentos em que a taxa de acerto foi de 0%. Este erro isolado se deve ao fato de a rede "confundir" um modelo específico de lâmpada fluorescente com um carregador, pois os resultados individuais de cada teste mostraram que o resultado desta análise sempre convergiram para classificar esta lâmpada como um carregador. Observando as formas de onda desta lâmpada (lâmpada fluorescente *Phillips*), e a dos carregadores, como pode ser visto nas Figuras 15 e 17, observa-se uma grande similaridade entre os dois tipos de sinais.

Para que se pudesse explorar a possibilidade de menor confusão entre a lâmpada fluorescente da marca *Phillips* e os carregadores para o caso de excluído, cogitou-se remover um dos quatro carregadores para testar a taxa de acerto desta lâmpada. Na prá-

tica, testou-se remover do grupo de treinamento as réplicas ruidosas de cada carregador, um por vez, assim, ao invés de 4 tipos de carregadores no treinamento, teríamos 3, sendo 30 réplicas ruidosas. Este processo foi realizado e repetido para cada exclusão de cada modelo de carregador. Observou-se que a confusão não se dava por um modelo específico, pois para os 4 testes, a lâmpada fluorescente *Phillips* apresentou uma taxa de acerto muito baixa. Para 3 dos modelos, os resultados continuaram como 0%, com a confusão ainda acontecendo para classificá-la como carregador. Para a exclusão do modelo do carregador genérico, a taxa de acerto foi de 10%, o que pode significar que este integrante é o principal responsável pela confusão entre as duas classes, mas não é o único. Assim, um último teste foi realizado, com a exclusão dos 4 carregadores do treinamento (exclusão das 40 réplicas), e o resultado foi 100% de acerto para todas as lâmpadas, incluindo a da marca *Phillips*.

Em um próximo projeto, este erro poderá ser minimizado utilizando-se de mais características para a definição de cada equipamento, como por exemplo a corrente de pico média. Assim, na entrada da rede neural, não serão introduzidas somente as componentes de frequência, mas também estas amplitudes de pico, o que poderá ajudar a definir e diferenciar ainda mais cada eletroeletrônico.

Figura 40: Taxa de acerto da rede convolucional - *Sigmoid*.

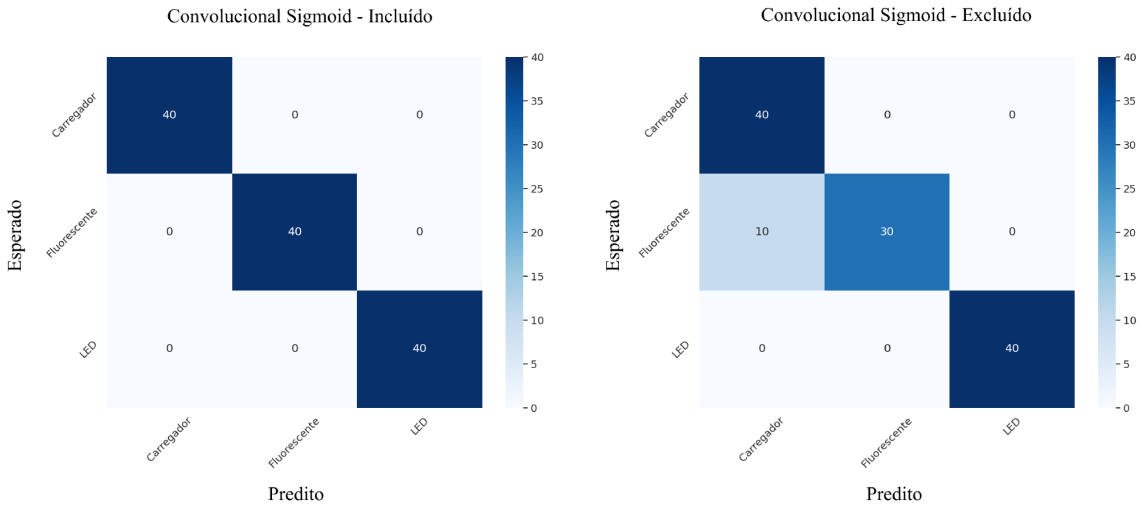


Fonte: O autor.

A Figura 41 nos mostra as matrizes de confusão para a rede convolucional com função *Sigmoid*. Podemos observar que para o caso de incluído somente a diagonal principal está preenchida, mostrando que não houve nenhuma confusão. Para o caso de excluído, observa-se que a confusão ocorrida com as lâmpadas fluorescentes, aconteceu 100% com a categoria dos carregadores.

Apesar deste resultado, em que se obteve uma taxa de acerto de 0% para uma lâmpada específica, o resultado geral obtido para este tipo de rede foi muito satisfatório, onde se obteve uma taxa de acerto geral de 100% para o caso de incluído, e de aproxima-

Figura 41: Matrizes de confusão da rede convolucional - *Sigmoid*.



Fonte: O autor.

damente 91% para o caso de excluído. Esta rede mostrou-se capaz de predizer qual tipo de eletrodoméstico está sendo analisado, mesmo sem conhecer previamente o seu sinal, para a maior parte dos equipamentos.

Para fins de comparação, calculou-se a taxa de acerto geral das três redes, e o resultado pode ser observado na Figura 42. Observa-se que a implementação com a rede neural convolucional e função de ativação *Sigmoid* nos neurônios da camada de entrada é a melhor para esta aplicação.

Figura 42: Taxa de acerto geral das três redes neurais.

Taxa de Acerto Geral		
Configuração da Rede	**Equipamento Incluído no Treinamento**	**Equipamento Excluído do Treinamento**
Não Convolucional	49,17%	37,50%
Convolucional ReLU	94,17%	77,50%
Convolucional Sigmoid	100,00%	91,67%

Fonte: O autor.

5 CONCLUSÃO

Como pôde ser observado nas seções anteriores, a rede neural corretamente configurada e os dados tratados da forma adequada, foram capazes de atingir o objetivo deste trabalho: classificar um eletrodoméstico com base no seu padrão de consumo de corrente elétrica. Observou-se que a rede convolucional com função de ativação *sigmoid* nos neurônios da camada de entrada resultou em uma melhor taxa de acerto quando comparada às outras implementações. Para que tal resultado pudesse ser atingido, foi

necessário coletar amostras de corrente de diferentes equipamentos, padronizar os dados por meio de uma série de processos e por fim submeter os dados a uma rede neural arquitetada para este tipo de situação. Observou-se que o processo como um todo tinha um melhor resultado quando o número de amostras utilizadas foi de 100, o número de réplicas foi 10, e a constante multiplicativa do ruído adicionado foi de 0,02. Utilizou-se a média móvel nas FFTs e o número de épocas foi de 10. A rede neural com função de ativação *sigmoid* nos neurônios da camada de entrada trouxe uma taxa de acerto de 100% quando incluídas no grupo de treinamento as réplicas ruidosas referentes ao equipamento a ser predito, e 91% quando não incluídas. As outras arquiteturas das RNAs foram a convolucional com função $ReLU$ na camada de entrada, e a não convolucional. A convolucional com função $ReLU$ apresentou 94,17% de taxa de acerto para o caso de incluído, e 77,50% para o caso de excluído. A rede não convolucional obteve 49,17% e 37,50% para os casos de incluído e excluído respectivamente. Pode-se observar que em todos os casos a rede tem uma taxa de acerto maior para o caso de incluído, mas ainda sim, conseguiu-se atingir resultados satisfatórios para o caso de excluído com a configuração convolucional com função *sigmoid*.

Para melhores resultados, o próximo passo seria a implementação de mais características dos eletrodomésticos no treinamento da rede neural, como por exemplo a corrente de pico média de cada aparelho.

Além de melhorar a eficácia do sistema em classificar eletrodomésticos quando analisados isoladamente, o avanço deste projeto se dará no sentido de reconhecer vários equipamentos ligados simultaneamente, onde o sistema deverá ser capaz de predizer quais os tipos de equipamentos estão ligados, utilizando somente um medidor de corrente. Com o primeiro passo já implementado (reconhecer o tipo de aparelho por meio do seu padrão de corrente), o segundo será o de entender quais as principais componentes de frequência (ou até mesmo outras características) são as mais importantes para se definir e classificar o tipo de um eletrodoméstico quando ligado em conjunto a outros. Assim, o projeto como um todo se aproximará do objetivo final, que é, reconhecer os dispositivos ligados ao mesmo tempo em uma casa, para entregar aos consumidores finais um relatório contendo os principais gastos com eletricidade. O intuito é que este relatório seja gerado com um baixo custo, visto que não serão necessários vários medidores para que se possa coletar um sinal para cada eletrodoméstico, e sim um único medidor para todos.

REFERÊNCIAS

ABRAHÃO, K. C. de F. J.; SOUZA, R. G. V. de. Estimativa da evolução do uso final de energia elétrica no setor residencial do brasil por região geográfica. *Ambiente Construído*, FapUNIFESP (SciELO), v. 21, n. 2, p. 383–408, abr. 2021. Disponível em: <https://doi.org/10.1590/s1678-86212021000200532>.

BERGER, C. *Perceptrons - the most basic form of a neural network*. 2016. Disponível em: <https://appliedgo.net/perceptron>. Acesso em: Janeiro 2024.

BOLETINS - 2009: 18 estados brasileiros e o Paraguai sofrem apagão após falha em Itaipu. 2016. Disponível em: <https://cbn.globoradio.globo.com/institucional/historia/aniversario/cbn-25-anos/boletins/2016/04/26/2009-18-ESTADOS-BRASILEIROS-E-O-PARAGUAI-SOFREM-APAGAO-APOS-FALHA-EM-ITAIPU.htm>. Acesso em: 21 set. 2023.

BORGES, F. Q. CRISE DE ENERGIA ELéTRICA NO BRASIL - UMA BREVE REFLEXão SOBRE a DINâMICA DE SUAS ORIGENS e RESULTADOS. *RECIMA21 - Revista Científica Multidisciplinar - ISSN 2675-6218*, RECIMA21 - Revista Cientifica Multidisciplinar, v. 2, n. 10, p. e210809, nov. 2021. Disponível em: <https://doi.org/10.47820/recima21.v2i10.809>.

BOYLESTAD, R. L. *Introdução à Análise de Circuitos*. 12.. ed. São Paulo: Pearson Prentice Hall, 2012. ISBN 978-85-64574-20-5.

EPE. *[Empresa de Pesquisa Energética] Balanço Energético Nacional 2023: Ano base 2022*. 2023. Disponível em: <https://www.epe.gov.br>. Acesso em: 15 set. 2023.

GÉRON, A. *Mãos à Obra Aprendizado de Máquina com Scikit-Learn & TensorFlow*. Rio de Janeiro: Alta Books, 2019. ISBN 978-85-508-0381-4.

GRUS, J. *Data Science do Zero*. Rio de Janeiro: Alta Books, 2016. ISBN 978-85-7608-998-8.

GUZMAN, L. G. *Deep neural networks, or Perceptron vs dogs and cats*. 2019. Disponível em: <https://inteligenciafutura.mx/english-version-blog/blog-06-english-version>. Acesso em: Novembro 2023.

KOUTROUMBAS, K.; THEODORIDIS, S. *Pattern Recognition*. 4.. ed. San Diego: Academic Press, 2009. ISBN 978-15-97492-72-0.

KUO, C. *What Is Image Recognition?* 2018. Disponível em: <https://medium.com/dataman-in-ai/module-6-image-recognition-for-insurance-claim-handling-part-i-a338d16c9de0>. Acesso em: Maio 2024.

LEITE, T. M. *Redes Neurais, Perceptron Multicamadas e o Algoritmo Backpropagation*. 2018. Disponível em: <https://medium.com/ensina-ai/redes-neurais-perceptron-multicamadas-e-o-algoritmo-backpropagation-eaf89778f5b8>. Acesso em: Maio 2024.

MALAR, J. P. *Crise energética deve aliviar em 2022, mas espaço para queda em contas é pequeno*. 2022. Disponível em: <https://www.cnnbrasil.com.br/economia/crise-energetica-deve-aliviar-em-2022-mas-espaco-para-queda-em-contas-e-pequeno>. Acesso em: 21 set. 2023.

MARTINS, R. *Racionamento atrapalhou retomada da economia em 2001; saiba se problema pode se repetir*. 2021. Disponível em: <https://g1.globo.com/economia/noticia/2021/06/29/racionamento-atrapalhou-retomada-da-economia-em-2001-saiba-se-problema-pode-se-repetir.ghtml>. Acesso em: 21 set. 2023.

NAVE, R. *Use of Complex Impedance*. 2000. Disponível em: <http://hyperphysics.phy-astr.gsu.edu/hbasees/electric/impcom.html>. Acesso em: Julho 2023.

OPPENHEIM A. V; WILLSKY, A. S. *Sinais e Sistemas*. 2.. ed. São Paulo: Prentice Hall, 2010. ISBN 978-85-7605-504-4.

SAHA, S. *A Comprehensive Guide to Convolutional Neural Networks the ELI5 way*. 2018. Disponível em: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>. Acesso em: Maio 2024.

SIDHARTH, G. *Multi-Layer Perceptron Explained: A Beginner's Guide*. 2023. Disponível em: <https://www.pycodemates.com/2023/01/multi-layer-perceptron-a-complete-overview.html>. Acesso em: Janeiro 2024.

STILPEN, D. V. de S.; CHENG, V. Solar photovoltaics in brazil: A promising renewable energy market. In: *2015 3rd International Renewable and Sustainable Energy Conference (IRSEC)*. IEEE, 2015. Disponível em: <https://doi.org/10.1109/irsec.2015.7455077>.

ZHANG, K. *Moving Averages*. 2021. Disponível em: <https://www.npmjs.com/package/moving-averages>. Acesso em: 04 out. 2023.