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

KAIO JUNIO GONÇALVES

AUTENTICAÇÃO DE DISPOSITIVOS DE COMUNICAÇÃO SEM FIO UTILIZANDO APRENDIZAGEM PROFUNDA

Patos de Minas - MG
2023

KAIO JUNIO GONÇALVES

AUTENTICAÇÃO DE DISPOSITIVOS DE COMUNICAÇÃO SEM FIO UTILIZANDO APRENDIZAGEM PROFUNDA

Trabalho de conclusão de curso apresentado à banca examinadora como requisito parcial à obtenção do grau de Bacharel 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
2023

KAIO JUNIO GONÇALVES

AUTENTICAÇÃO DE DISPOSITIVOS DE COMUNICAÇÃO SEM FIO UTILIZANDO APRENDIZAGEM PROFUNDA

Trabalho de conclusão aprovado para obtenção do grau de Bacharel 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, 26 de Junho de 2023

Banca Examinadora

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

Prof. Dra. Karine Barbosa Carbonaro – FEELT/UFU (Membro 1)

Prof. Dr. Laurence Rodrigues do Amaral – FACOM/UFU (Membro 2)

RESUMO

Dispositivos eletrônicos de transmissão sem fio imprimem variações físicas únicas nos sinais que emitem, as quais podem ser utilizadas como uma assinatura ou impressão digital do dispositivo emissor. Essas variações podem ser interpretadas como ruídos que os componentes eletrônicos do transmissor imprimem no sinal. Devido a sutis variações no processo de fabricação, nenhum componente é idêntico a outro.

Para realizar o estudo, foram coletados sinais emitidos por dispositivos de comunicação sem fio de baixo custo, utilizando um equipamento de Rádio Definido por Software (SDR). Esses sinais foram processados para criar uma base de dados que foi utilizada no treinamento e validação de uma Rede Neural Convolucional (CNN). A CNN desenvolvida demonstrou uma acurácia de 98% na identificação dos dispositivos, utilizando apenas os sinais emitidos por eles.

A tecnologia proposta funciona de forma passiva, podendo ser implementada nos elementos de controle e de *gateway* de uma rede de dados para autenticação de dispositivos. Não é necessário alterar a programação dos demais elementos de borda da rede, que geralmente sofrem de severas limitações de hardware.

As redes neurais convolucionais provaram ser uma ferramenta poderosa e promissora, com capacidade de identificar padrões únicos em sinais de radiofrequência e diferenciar inequivocamente um dispositivo de outro.

Palavras-chave: rádio definido por software. redes neurais. aprendizado de máquina. aprendizagem profunda. rede neural convolunional. Internet das Coisas. comunicação sem fio.

ABSTRACT

Electronic wireless devices cause unique physics variations in the emitting signals that can be used as a signature or a fingerprint. These variations can be interpreted as noise introduced by the electronic components of the transmitter into the signal. Due to subtle variations in the manufacturing process, no component is identical to another.

To conduct the study, signals emitted by low-cost wireless communication devices were collected using a Software-Defined Radio (SDR) equipment. These signals were processed to create a database that was used for training and validation of a Convolutional Neural Network (CNN). The developed CNN demonstrated a 98% accuracy in device identification using only the signals emitted by them.

The proposed technology operates passively and can be implemented in the control and gateway elements of a data network for device authentication. There is no need to modify the programming of the other network edge elements, which often suffer from severe hardware limitations.

Convolutional neural networks have proven to be a powerful and promising tool with the ability to identify unique patterns in radiofrequency signals and unequivocally differentiate one device from another.

Keywords: Software Defined Radio. neural networks. machine learning. deep learning. convolutional neural network. Internet of Things. wireless communication.

LISTA DE FIGURAS

LISTA DE ABREVIAÇÕES

AM Aprendizado de Máquina.

CNN Rede Neural Convolucional.

DNN *Deep Neural Network*.

I/Q in-phase/quadrature.

IoE *Internet of Everythings*.

IoT *Internet of Things*.

ISM *Industrial Sientific and Medical*.

LTU *Linear Threshold Unit*.

ML *Machine Learning*.

MLP Perceptron Multicamada.

OTP *One-Time Pad*.

PDI Processamento Digital de Imagens.

PReLU Unidade Linear Retificada Parametrizada.

ReLU Unidade Linear Retificada.

RF sinais de radiofrequência.

RF-DNA *radio-frequency distinct native attribute*.

RNAs Redes Neurais Artificiais.

SDR Rádio Definido por Software.

TICs Tecnologias de Informação e Comunicação.

SUMÁRIO

1 INTRODUÇÃO

O mundo está cada vez mais interligado. A evolução da Internet tem provocado significativas mudanças na forma como nos conectamos. As conexões em rede cada vez mais envolvem pessoas, processos, dados e objetos. Transformam informações em ações, criando novos recursos, experiências mais ricas e oportunidades econômicas sem precedentes para empresas, indivíduos e países. Essa conectividade é chamada de Internet de Todas as Coisas ou *Internet of Everythings* (IoE) (CISCO, 2013).

Estima-se que em 2020 teremos no mundo cerca de 50 bilhões de dispositivos conectados (EVANS, 2012). A Internet de Todas as Coisas poderá movimentar um mercado de 14,4 trilhões de dólares entre 2013 e 2022 (BRADLEY; BARBIER; HANDLER, 2013). Diante desse cenário, em 2019 foi criado no Brasil o Plano Nacional de Internet das Coisas. Uma de suas definições é que a Internet das Coisas, ou *Internet of Things* (IoT), são sistemas de comunicação máquina a máquina (BRASIL, 2019). A IoT representa uma tecnologia transitória para a IoE, uma vez que não engloba pessoas e processos em suas redes de comunicação (CISCO, 2013). O Plano traz como prioridades as aplicações nas áreas de agronegócios, saúde, cidades inteligentes e indústria, além de reforçar a preocupação com a segurança e a privacidade (BRASIL, 2019).

À medida que a IoT emerge como uma das principais tendências que moldam o desenvolvimento das Tecnologias de Informação e Comunicação (TICs) deve-se aumentar a preocupação com as possíveis vulnerabilidades (BRADLEY; BARBIER; HANDLER, 2013). O ecossistema da Internet das Coisas é composto, em boa parte, pelos chamados dispositivos inteligentes. São dispositivos pequenos e altamente restritos em termos de memória, poder de processamento, autonomia de bateria e capacidade de comunicação (JARA; LADID; GÓMEZ-SKARMETA, 2013). Tais restrições expõem os dispositivos de IoT a uma série de vulnerabilidades e com isso a Internet das Coisas exige soluções de segurança feitas sob medida (LEO et al., 2014) (KEOH; KUMAR; TSCHOFENIG, 2014). Medidas de segurança tradicionais não se adequam diretamente à IoT devido aos diferentes padrões envolvidos (TEIXEIRA et al., 2015).

Nesse contexto, esse trabalho visa explorar a viabilidade de um sistema de autenticação que identifique inequivocamente um transmissor, utilizando-se das características físicas únicas que componentes eletrônicos imprimem em suas ondas de radiofrequência, como uma espécie de "impressão digital" do transmissor, a fim de fornecer uma camada adicional de segurança para dispositivos sem fio de baixo custo.

1.1 TEMA DO PROJETO

Esse trabalho visa explorar a viabilidade de empregar equipamento de Rádio Definido por *Software* (SDR) juntamente com a utilização de aprendizagem profunda

(*Deep Learning*) para autenticação de dispositivos de comunicação sem fio de baixo custo, geralmente empregados em Internet das coisas (IoT).

1.2 PROBLEMATIZAÇÃO

Atualmente os problemas de segurança em IoT têm se tornado mais e mais preocupantes devido ao crescente número de ataques. As redes IoT são mais vulneráveis que as redes tradicionais por causa das características dos dispositivos IoT e de seus protocolos de comunicação. Todos os elementos de uma rede IoT são potenciais alvos de ataques (CUI et al., 2018).

Dispositivos de IoT encontram-se presentes em residências, automação industrial e infraestrutura de cidades inteligentes. Estão o tempo todo produzindo uma quantidade maciça de dados e, por estarem interconectados com a Internet, podem ser acessados e gerenciados a qualquer momento e em qualquer lugar (KEOH; KUMAR; TSCHOFENIG, 2014).

As limitações em termos de performance e de *hardware* presentes nesses dispositivos dificultam padronizações e implementação de medidas de segurança eficientes (TEIXEIRA et al., 2015).

Diante desse cenário verifica-se a necessidade da criação de medidas de segurança que se apliquem a dispositivos de baixo custo e que possam contribuir para a segurança da rede IoT.

1.3 HIPÓTESES

Dispositivos eletrônicos de transmissão sem fio imprimem variações físicas únicas nos sinais que emitem. Essas variações podem ser utilizadas como uma assinatura ou "impressão digital" do dispositivo emissor.

Esse ruído, ou variações únicas, presentes nos sinais emitidos não sofrem modificações significativas ao longo da vida útil do equipamento transmissor.

Uma rede neural profunda será capaz de identificar esses padrões únicos e diferenciar inequivocamente um dispositivo de outro.

Utilizando somente um equipamento de rádio definido por *software* de baixo custo será possível captar os sinais transmitidos juntamente com os padrões únicos emitidos pelos transmissores.

1.4 OBJETIVOS

Esse trabalho tem como objetivo geral explorar a viabilidade de um sistema de autenticação que identifique inequivocamente um transmissor, através de suas ondas de

radiofrequência, a fim de fornecer uma camada adicional de segurança para dispositivos sem fio de baixo custo.

Os objetivos específicos são:

- Amostrar sinais de radiofrequência utilizando um equipamento de rádio definido por *software* (modelo RTLSDR RTL2832U DVB-T) e transmissores sem fio de baixo custo (controles remotos geralmente utilizados em portões de garagem);
- Construir uma base de dados de sinais de radiofrequência;
- Construir uma rede neural de aprendizagem profunda e treiná-la utilizando a base de dados de radiofrequência;
- Utilizar a rede neural profunda para autenticar os dispositivos de transmissão para os quais ela foi treinada.

1.5 JUSTIFICATIVAS

Um sistema de autenticação capaz de identificar elementos de uma rede IoT utilizando-se apenas de características físicas dos sinais recebidos pode representar um importante mecanismo de segurança.

Os elementos de borda de uma rede IoT geralmente enfrentam severas limitações de *hardware* e energia, além de possuírem diversos padrões de programação envolvidos (TEIXEIRA et al., 2015). Esses problemas podem ser contornados com o uso da tecnologia proposta, uma vez que não obrigatoriamente tal mecanismo de autenticação deva estar presente nos elementos de borda da rede IoT. A tecnologia proposta não necessita dos mecanismos convencionais de autenticação onde é preciso que transmissores e receptores continuamente troquem informações. Funcionando de forma passiva, o sistema de autenticação proposto não implica alterar a programação dos dispositivos de borda da rede, podendo ser implementado somente nos elementos de controle e de *gateway*, os quais não sofrem das mesmas limitações que os demais dispositivos da rede.

2 DESENVOLVIMENTO TEÓRICO

Os sinais de radiofrequência (RF) possuem atributos físicos intrínsecos distintos chamados *radio-frequency distinct native attribute* (RF-DNA). Características únicas, inerentes aos componentes eletrônicos, levam a alterações nos sinais por eles transmitidos, formando um padrão distinto de ruído que age como um tipo de impressão digital do dispositivo (WANG et al., 2019). Tais características diferem mesmo entre dispositivos de igual modelo e fabricante e se devem a pequenas imperfeições no processo de fabricação (PATEL; TEMPLE; BALDWIN, 2015).

Esses padrões podem ser utilizados como uma impressão digital para identificar ou autenticar inequivocamente um dispositivo em uma rede sem fio. Dessa forma é possível se prevenir de uma brecha bastante explorada para quebra de segurança, que é a clonagem de dispositivos da rede (WANG et al., 2019).

Para a análise e identificação desses padrões, primeiramente é preciso coletar os sinais emitidos pelos transmissores. Para isso, escolheu-se um equipamento de rádio definido por *software*.

2.1 RÁDIO DEFINIDO POR *SOFTWARE*

Os Rádios Definidos por *Software* ou SDR são um conjunto de tecnologias que envolvem *hardware e software*, de tal forma que, funções antes implementadas em *hardware* passaram a ficar a cargo do *software*, tornando-os mais flexíveis e fáceis de serem programados. São transceptores sem fio capazes de operar em múltiplas frequências e capazes de se adaptar, via *software*, às diversas codificações e tecnologias de comunicação existentes (SILVA et al., 2015).

Os SDRs são aplicáveis em diversas áreas como radioamadorismo, pesquisa experimental e também profissional, em redes de computadores e telecomunicações (SILVA et al., 2015).

Muitos entusiastas acabaram focando em produzir seus próprios receptores para aplicações personalizadas. Produtos comerciais inicialmente não projetados como SDRs acabaram sendo reprogramados para essa finalidade. Usuários descobriram que alguns *hardwares* realizavam todo o processamento de sinais em *software* (CASS, 2013).

Em uma dessas iniciativas, desenvolvedores descobriram que era possível usar *dongles* de baixo custo que possuem o controlador Realtek RTL2832U para criar um SDR capaz de receber frequências, na faixa de 500 kHz até 1,75 GHz, dependendo do tipo de antena empregada. Surgiu então, como resultado de um esforço coletivo, o RTL-SDR, *hardware* de aproximadamente vinte e cinco dólares que usa *software* livre, capaz de escanear sinais de rádio em tempo real (RTL-SDR.COM, 2019).

Com o passar dos anos desde sua descoberta, o RTL-SDR se tornou bastante

popular e tem democratizado o acesso ao espectro eletromagnético. Por todas as características apresentadas, escolheu-se o modelo RTLSDR RTL2832U DVB-T Tuner Dongle para a coleta de dados dos sinais dos transmissores. Os dados são coletados através de amostras in-phase/quadrature (I/Q) em que (I) representa a fase do sinal e é expresso por um número real e (Q) representa a quadratura do sinal e é expresso por um número imaginário.

Utilizando aprendizado de máquina é possível treinar uma rede neural artificial com as amostras I/Q para que ela possa diferenciar um transmissor de outro através de padrões únicos nos sinais.

2.2 APRENDIZADO DE MÁQUINA

Aprendizado de Máquina (AM) ou *Machine Learning* (ML) é o campo de estudos que dá aos computadores a habilidade de aprenderem sem serem explicitamente programados (SAMUEL, 1959). Pode ser entendida também como a ciência da programação de computadores para a aprendizagem com dados (GÉRON, 2019).

O AM se destaca na solução de problemas nos quais as soluções tradicionais exigem muita configuração manual ou extensas listas de regras. O algoritmo de AM simplifica e melhora o código. Outro ponto de destaque é na solução de problemas complexos nos quais não há uma solução aceitável quando utilizada uma abordagem tradicional. Nesses casos, técnicas sofisticadas de AM podem encontrar uma solução. Em aplicações com ambientes variáveis que exigem adaptação a novos dados o AM também se sobressai, além de auxiliar na compreensão de problemas complexos e grande volume de dados (GÉRON, 2019).

Existem diferentes tipos de sistemas de Aprendizado de Máquina que podem ser classificados com base em várias características. Uns podem ser treinados com ou sem supervisão humana. Outros podem ou não aprender rapidamente de forma incremental. Além disso, alguns sistemas funcionam comparando novos pontos de dados com pontos conhecidos, enquanto outros detectam padrões nos dados de treinamento para criar um modelo preditivo, semelhante ao trabalho dos cientistas. É importante ressaltar que esses critérios não são mutuamente exclusivos e podem ser combinados de várias maneiras (GÉRON, 2019). No caso deste trabalho em específico, estaremos trabalhando com redes neurais artificiais.

2.3 REDES NEURAIS ARTIFICIAIS

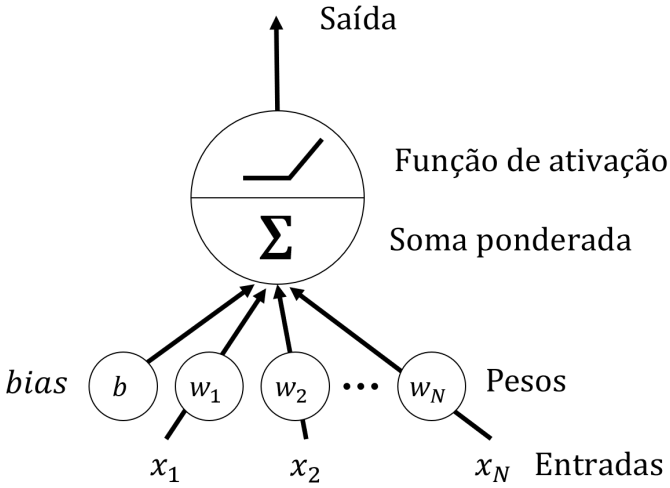
As Redes Neurais Artificiais (RNAs) são uma técnica de Aprendizado de Máquina. São inspiradas na arquitetura do cérebro e em neurônios biológicos. No entanto, a analogia serve apenas de inspiração, uma vez que as RNAs têm tomado caminhos diferentes dos biológicos. As RNAs são aplicações versáteis, poderosas e escaláveis, o que as

tornam ideais para tarefas complexas como classificar bilhões de imagens, prover serviços de reconhecimento de fala, recomendar resultados de pesquisas na Internet ou derrotar campeões mundiais em jogos (GÉRON, 2019).

O surgimento das RNAs se deu em 1943, com o trabalho do neurofisiologista Warren McCulloch e do matemático Walter Pitts (MCCULLOCH, 1943). Desde então, diversos modelos foram propostos até as atuais redes neurais.

O neurônio artificial modificado utilizado é chamado de unidade linear com *threshold* ou *Linear Threshold Unit* (LTU) e pode ser visto na Figura 1:

Figura 1: LTU - Unidade Linear com *Threshold*



Fonte: Adaptado de (GÉRON, 2019, p.263).

Primeiramente, é realizada uma soma ponderada com os sinais de entrada da LTU, representados pelo vetor $\boldsymbol{X} = [x_1, x_2, ...x_n]$, multiplicados pelos respectivos pesos sinápticos do vetor $\boldsymbol{W} = [w_1, w_2, ...w_n]$. O termo adicional b também entra no somatório e representa o viés (*bias*), ou desvio, um termo independente que não está associado a nenhuma das variáveis de entrada. Em seguida, é aplicada uma função de ativação. Trata-se de uma função matemática que têm por finalidade determinar uma saída limitada para o neurônio em função da soma ponderada de suas entradas. As funções de ativação introduzem um componente não linear nas redes neurais, que faz com que elas possam aprender mais do que relações lineares entre as variáveis dependentes e independentes. Existem diversas funções de ativação: Sigmoide, Tangente Hiperbólica, Degrau, ReLU, PReLU dentre outras (GÉRON, 2019). A Unidade Linear Retificada (ReLU) é uma das funções de ativação mais amplamente utilizada atualmente e é definida por:

$$ReLU(x) = max\{0, x\} \tag{1}$$

Essa função tem saída zero para qualquer entrada negativa e saída igual a entrada para qualquer valor de entrada positivo. Uma variante da ReLU é a Unidade

Linear Retificada Parametrizada (PReLU).

$$PReLU(x) = max\{0, x\} + a.min\{0, x\} \tag{2}$$

em que x é a entrada do neurônio e a é um parâmetro aprendível. A PReLU tem saída igual a entrada para qualquer valor de entrada positivo e para valores de entrada negativos a saída é igual a entrada multiplicado por um peso a, o qual é ajustado a cada iteração (HE et al., 2015).

O modelo matemático do neurônio pode ser representado então por:

$$y = \varphi(\sum_{n=1}^{N} x_n w_n + b) \tag{3}$$

ou na forma matricial por:

$$y = \varphi(\boldsymbol{W}\boldsymbol{X} + b) \tag{4}$$

em que y é a saída do neurônio e φ representa a função de ativação.

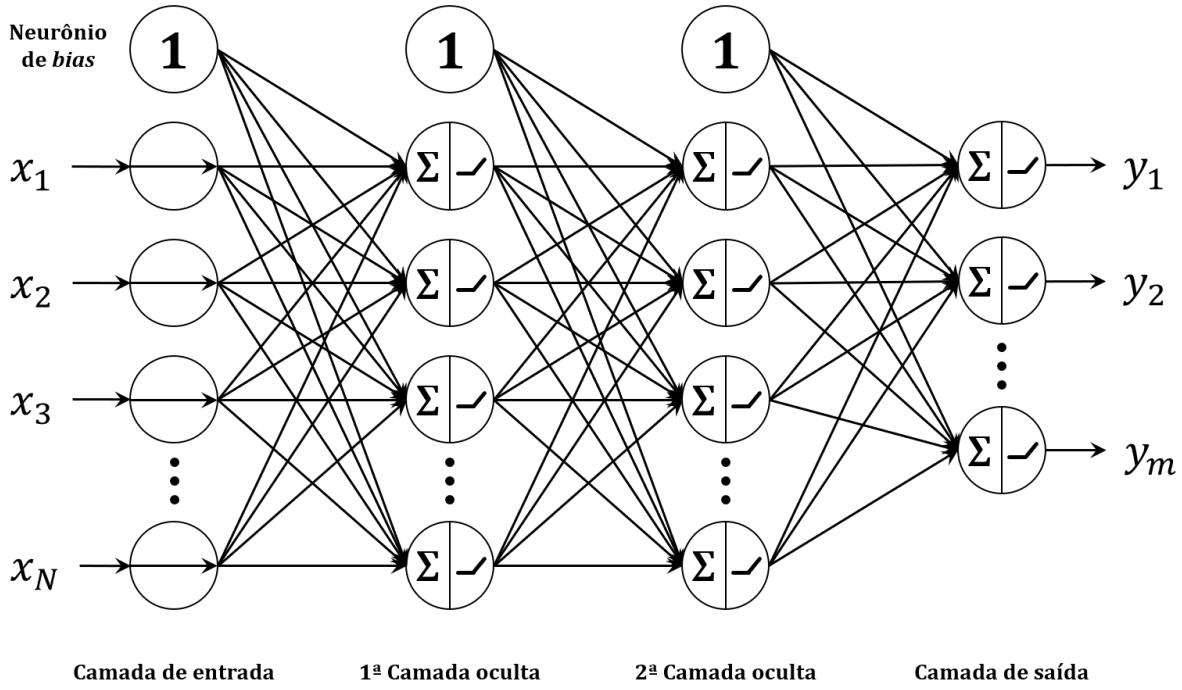
Uma arquitetura de rede neural artificial baseada em LTUs é chamada de Perceptron. Quando temos uma rede com uma camada de entrada, uma ou mais camadas LTUs ocultas e uma camada final LTU de saída, damos o nome de Perceptron Multicamada (MLP). Se a rede neural artificial possui duas ou mais camadas ocultas também é chamada de Rede Neural Profunda ou *Deep Neural Network* (DNN). O ramo do AM que trata das redes neurais profundas é também chamado de aprendizagem profunda ou *deep learning* (GÉRON, 2019).

A Figura 2 representa uma Rede Neural Profunda com suas camadas. A primeira é a camada de entrada, composta de neurônios sensoriais que podem corresponder aos *pixels* de uma imagem, por exemplo. As camadas ocultas são compostas pelos neurônios de processamento e sua quantidade torna a rede capaz de aprender relações cada vez mais complexas. Por fim, temos a camada de saída onde os neurônios representam as possíveis respostas da rede. Um modelo matemático geral de uma DNN com duas camadas ocultas pode ser expresso na forma matricial por:

$$y = \varphi_3(\boldsymbol{W}_3.\varphi_2(\boldsymbol{W}_2.\varphi_1(\boldsymbol{W}_1.\boldsymbol{X} + b_1) + b_2) + b_3) \tag{5}$$

Em que y é a saída da rede; φ_1, φ_2 e φ_3, representam respectivamente as funções de ativação da primeira camada oculta, segunda camada oculta e camada de saída; $\boldsymbol{W}_1, \boldsymbol{W}_2$ e $\boldsymbol{W}_3$ representam respectivamente as matrizes de pesos da primeira camada oculta, segunda camada oculta e camada de saída; b_1, b_2 e b_3 representam respectivamente os neurônios de *bias* da camada de entrada, da primeira camada oculta e da segunda camada oculta e, por fim, $\boldsymbol{X}$ representa a matriz das variáveis de entrada (CASTRO; ZUBEN, 2003).

Figura 2: Rede Neural Profunda



Fonte: Adaptado de (CASTRO; ZUBEN, 2003, p.55)

Para que uma rede dessas funcione é preciso treiná-la. O treinamento de uma rede MLP insere-se no contexto de aprendizagem de máquina supervisionado, em que cada amostra de dados utilizada é associada a um rótulo informando a que classificação ela pertence. A ideia geral é fazer com que a rede aprenda padrões para que quando for fornecida uma amostra desconhecida a rede seja capaz de dizer a qual classe a amostra pertence (LEITE, 2018).

Durante muitos anos os pesquisadores procuraram uma forma de treinar MLPs, mas só em 1985 Rumelhart et al. conseguiram apresentar um algoritmo inovador de treinamento, a retropropagação (RUMELHART; HINTON; WILLIAMS, 1985).

Quando criamos nossas redes neurais não sabemos quais valores ideais escolher para os pesos e vieses (*bias*). Por isso, geralmente atribuímos valores aleatórios que provavelmente ocasionarão erros na saída da rede. A retropropagação, ou *backpropagation*, baseia-se no erro ocorrido na camada de saída da rede neural para recalcular o valor dos pesos do vetor $\boldsymbol{W}$. Detectado o erro na saída recalcula-se o valor de todos os pesos, começando da última camada e indo até a primeira, sempre tendo em vista diminuir o erro (LEITE, 2018).

O erro da saída pode ser calculado através da somatória dos quadrados das diferenças entre a saída esperada e a saída obtida pela rede:

$$E(y, \hat{y}) = (\sum_{i=1}^{N} y_i - \hat{y}_i)^2 \tag{6}$$

Em que E é o erro da saída, y é a saída esperada e $\hat{y}$ é a saída obtida da rede. Lembrando que como o aprendizado é supervisionado, já se sabe de antemão qual deveria ser a resposta correta.

Para minimizar o valor da função de erro, calcula-se os valores dos gradientes para cada peso da rede. No cálculo vetorial o gradiente é um vetor que indica a direção de maior crescimento de uma função. Aqui, basta inverter o sinal para decrescer os resultados da função erro. A atualização dos pesos é feita de modo iterativo utilizando a seguinte fórmula de atualização dos pesos:

$$w_{atual} = w_{anterior} - \eta \frac{\partial E}{\partial w} \tag{7}$$

Em que o valor do peso na iteração atual será o valor do peso na iteração anterior, corrigido de valor proporcional ao gradiente. O sinal negativo indica que estamos indo na direção contrária à do gradiente para que ocorra o decréscimo do erro. O parâmetro η representa a taxa de aprendizado da rede neural e pode ser definido pelo criador da rede para controlar o tamanho do passo tomado na correção do peso. Passos muito grandes podem fazer com que a correção dos erros pule os valores de w que levam na direção de um erro mínimo e passos muito curtos podem aumentar significativamente o número de iterações para se chegar ao menor valor do erro (LEITE, 2018).

O termo chave da equação 7 é o cálculo da expressão $\partial E/\partial w$, consistindo em calcular as derivadas parciais da função de erro E em relação a cada peso do vetor w. Por isso é importante que a função de erro seja diferenciável. Da equação 6 sabemos que a função erro depende da função y, que por sua vez, de acordo com a equação 5, depende da função de ativação. Por isso a função de ativação também deve ser diferenciável.

Cumpridos todos esses requisitos é possível calcular as derivadas parciais da função de erro E que é uma função composta de outras funções. Em cálculo utiliza-se a regra da cadeia para derivar esse tipo de função:

$$\frac{d}{dx}[f(g(x))] = f'(g(x)).g'(x) \quad ou \quad \frac{df}{dx} = \frac{df}{dg}.\frac{dg}{dx} \tag{8}$$

As demonstrações matemáticas da aplicação da regra da cadeia, Eq. 8, na função erro E, Eq. 6, fogem ao propósito do trabalho e podem ser encontradas nas referências utilizadas.

Com o algoritmo de retropropagação tornou-se possível treinar uma rede neural de forma que os pesos são ajustados até que a função erro alcance os valores mínimos. Os neurônios de saída da rede correspondem às classes para as quais a rede foi treinada. Uma DNN com dois neurônios na camada de saída, por exemplo, pode ser usada para

diferenciar imagens de cães e gatos, onde temos um neurônio de saída para cada tipo de animal (CASTRO; ZUBEN, 2003).

Quando as classes da rede são mutualmente exclusivas, as funções de ativação individuais de cada neurônio da camada de saída são substituídas por uma função de ativação compartilhada chamada de Softmax. Essa função tem por finalidade oferecer uma saída normalizada, limitada entre os valores zero e um, que resulta na probabilidade individual de cada neurônio de saída em ser a resposta da rede (GÉRON, 2019).

Modelos de aprendizagem profunda mais atuais como as Redes Neurais Convolucionais ou *Convolutional Neural network* (CNN), embora mais refinados que as MLPs também utilizam como método de aprendizado a retropropagação. As CNNs conseguem encontrar padrões inobserváveis e obscuros a nós humanos (LEITE, 2018).

2.4 REDES NEURAIS CONVOLUCIONAIS

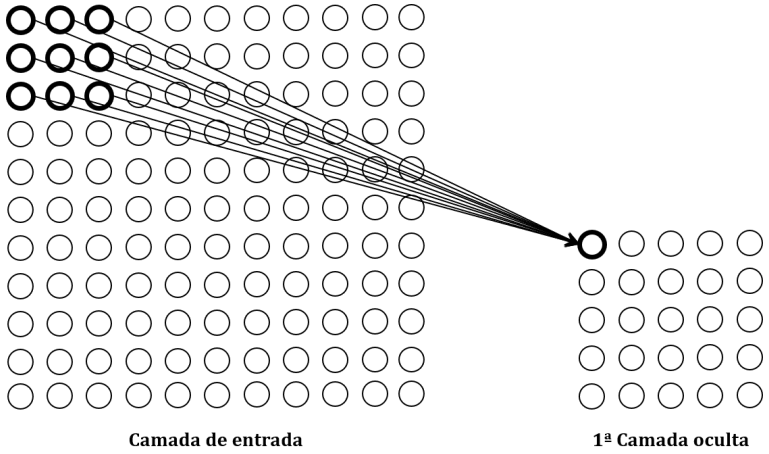
As Redes Neurais Convolucionais (CNN) emergiram do estudo do córtex visual do cérebro. Os ganhadores do Prêmio Nobel de Fisiologia e Medicina de 1981, David H. Hubel e Torsten Wiesel, mostraram em seu premiado trabalho que muitos neurônios no córtex visual têm um pequeno campo receptivo local. Isso significa que há grupos de neurônios especializados em reagir somente a determinado estímulo no campo de visão. Para isso os neurônios não se conectam a todos os outros como nos MLPs; cada neurônio está conectado a apenas alguns neurônios da camada anterior. Trata-se de uma poderosa arquitetura capaz de detectar todos os tipos de padrões complexos no campo visual. Além disso as CNNs não estão restritas ao campo visual, sendo também bem-sucedidas no reconhecimento de voz e no processamento de linguagem natural (GÉRON, 2019).

Valendo-se da analogia biológica Lecun et al. (1998) publicou a famosa arquitetura LeNet-5, a qual adicionava duas novas camadas às já utilizadas nas MLPs: as camadas convolucionais e as camadas de *pooling* (LECUN et al., 1998).

A camada convolucional é o bloco de construção mais importante da CNN. Os neurônios na primeira camada não estão conectados a cada *pixel* da imagem de entrada como acontecia nos Perceptrons Multicamada, mas apenas a *pixels* em seus campos receptivos. Um campo receptivo é uma área na imagem de entrada, geralmente um retângulo, com dimensões de largura e altura reduzidas. Cada neurônio na segunda camada convolucional está conectado somente aos neurônios de seu respectivo retângulo da primeira camada, como pode se ver na Figura 3. Essa arquitetura permite que a rede se concentre em características de baixo nível na primeira camada oculta e as agrupe em características de nível superior na próxima camada, e assim por diante.

Ocorre mudança também na entrada da rede que, anteriormente nos MLPs, se dava na forma de um vetor unidimensional e agora cada camada é representada por uma matriz de duas dimensões (GÉRON, 2019).

Figura 3: Campo receptivo de área 3x3 em uma entrada de 11x11 neurônios



Fonte: o autor.

Os pesos das conexões dos neurônios no campo receptivo formam, na prática, matrizes de convolução, as mesmas utilizadas em Processamento Digital de Imagens (PDI) para aplicação de filtros capazes de detecção de bordas, relevo, nitidez, desfocagem, dentre vários outros. A única diferença é que os valores ou pesos das matrizes de convolução não são pré-determinados como em PDI. São iniciados aleatoriamente e depois vão se ajustando através do processo de retropropagação. Essa matriz de convolução, formada pelos pesos das conexões dos neurônios do campo receptivo, é chamada de filtro, máscara ou *kernel* de convolução (ALVES, 2018).

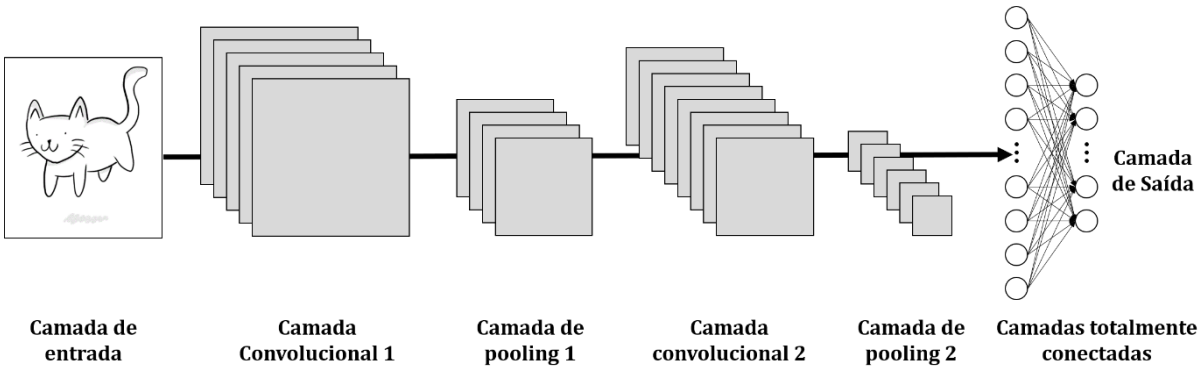
A convolução é uma operação matemática que envolve apenas multiplicações e somas. No nosso caso, com matrizes, basta multiplicar cada elemento da matriz do campo receptivo pelo respectivo peso na matriz do *kernel* e somar todos os resultados. Quanto maior o resultado, maior é a compatibilidade entre o filtro e a região analisada. Após isso, o resultado passa pela função de ativação de forma análoga aos MLPs. Para completar o processo é necessário "escorregar", processo pelo qual se varre toda a imagem de entrada com os campos receptivos para capturar os traços mais marcantes. O resultado é uma outra matriz com dimensionalidade reduzida, chamada de mapa de características ou *feature map*, que na Figura 3 corresponde à 1ª camada oculta. É comum adicionar zeros às bordas da entrada, aumentando sua dimensão, para que, após a convolução, a camada seguinte tenha o mesmo tamanho da entrada original. Esse processo é chamado de *zero padding* e serve para que as camadas não diminuam muito mais rápido do que é necessário para o aprendizado (ALVES, 2018)(GÉRON, 2019).

Para varrer toda a figura com o retângulo do campo receptivo é preciso determinar a distância entre dois campos receptivos consecutivos. Essa distância é chamada de *stride*, e pode ser vertical e horizontal. Os campos receptivos podem se sobrepor, ter tamanhos diferentes e também ter *strides* verticais e horizontais diferentes (GÉRON,

2019). Na Figura 3, por exemplo, temos um campo receptivo de 3x3 que, se utilizado com um *stride* vertical e horizontal de tamanho dois, pode varrer a entrada 11x11 e obter um *feature map* de 5x5 neurônios.

Uma camada convolucional, na verdade, aplica simultaneamente vários filtros às suas entradas, tornando-a capaz de detectar várias características. Isso quer dizer que cada camada convolucional é composta de inúmeros mapas de características, um para cada filtro, conforme pode ser visto na Figura 4, que representa a arquitetura típica de uma Rede Neural Convolucional. A quantidade de filtros deve ser definida pelo criador da rede (GÉRON, 2019).

Figura 4: Arquitetura típica de uma CNN



Fonte: Adaptado de (GÉRON, 2019, p.381)

É possível observar na Figura 4 a presença das camadas de *pooling*. Seu objetivo é subamostrar, ou seja, reduzir as dimensões da imagem para diminuir o custo computacional. Como nas camadas convolucionais, cada neurônio na camada *pooling* é conectado a um número limitado de neurônios na camada anterior, no campo receptivo retangular. Da mesma forma deve-se definir o tamanho e o *stride* do *kernel*. A diferença aqui é que um neurônio de *pooling* não possui pesos associados, ele apenas une as entradas usando a função máximo ou a média. A camada *pooling* mais comum é a do tipo *max pooling*. Ela apenas retorna o maior valor dentre os números da matriz do *pooling kernel* aplicada na camada anterior (GÉRON, 2019).

A arquitetura típica de uma Rede Neural Convolucional, apresentada na Figura 4, empilha alguns mapas de características na camada convolucional, depois uma camada *pooling*, depois outra camada convolucional, outra *pooling*, e assim por diante. A dimensão da imagem de entrada fica cada vez menor à medida que a rede avança, mas se torna mais profunda, com mais mapas de características, devido às camadas convolucionais. Na parte final é adicionada uma rede neural MLP convencional, com camadas totalmente conectadas, e a camada de saída com a função Softmax gera as previsões (GÉRON, 2019).

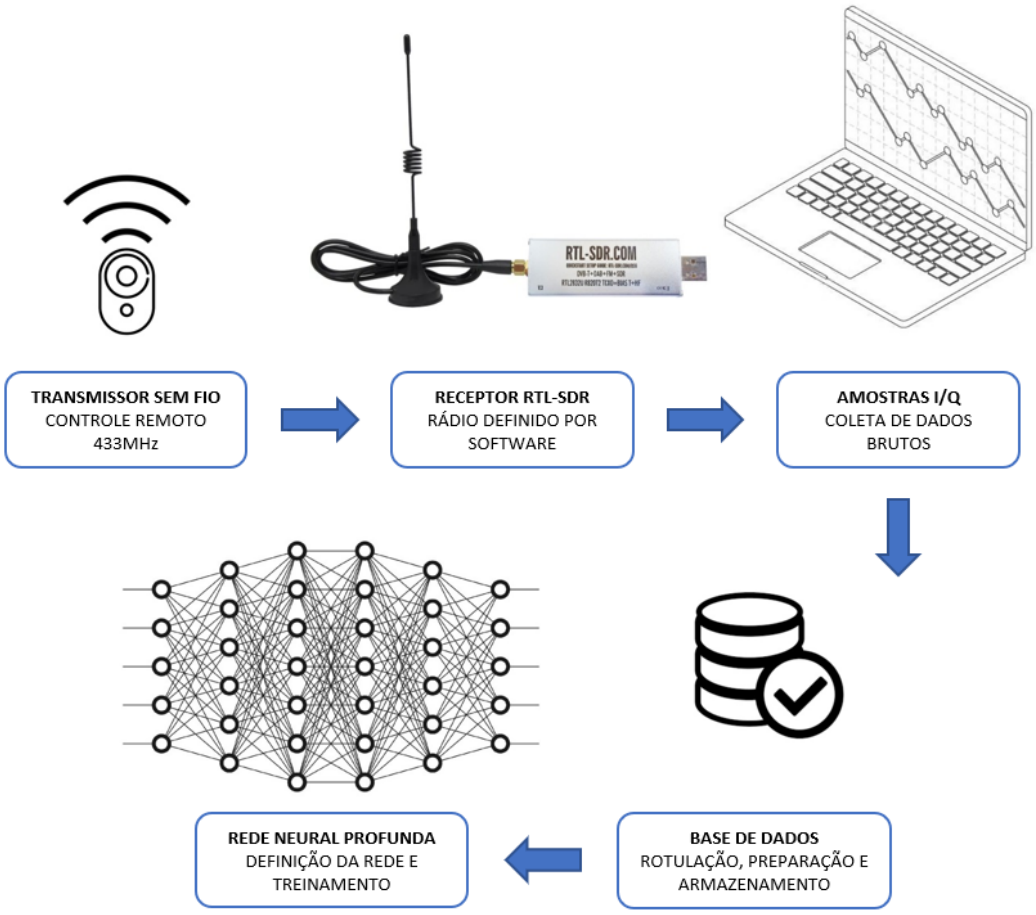
Conforme discutido anteriormente as Redes Neurais Convolucionais não estão restritas ao campo visual. Estudos recentes demonstraram que essas redes são eficientes

também na identificação de padrões de ruído emitidos por transmissores sem fio (RIYAZ et al., 2018).

3 MATERIAIS E MÉTODOS

O esquema proposto para a implementação da pesquisa está representado na Figura 5. Ele consiste em utilizar os controles para emissão dos sinais. Em seguida, os sinais são coletados por meio de um equipamento de rádio definido por *software*. O próximo passo é realizar o tratamento desses sinais e armazená-los como uma base de dados para o treinamento da rede. Por fim, a rede é treinada utilizando a base de dados coletada.

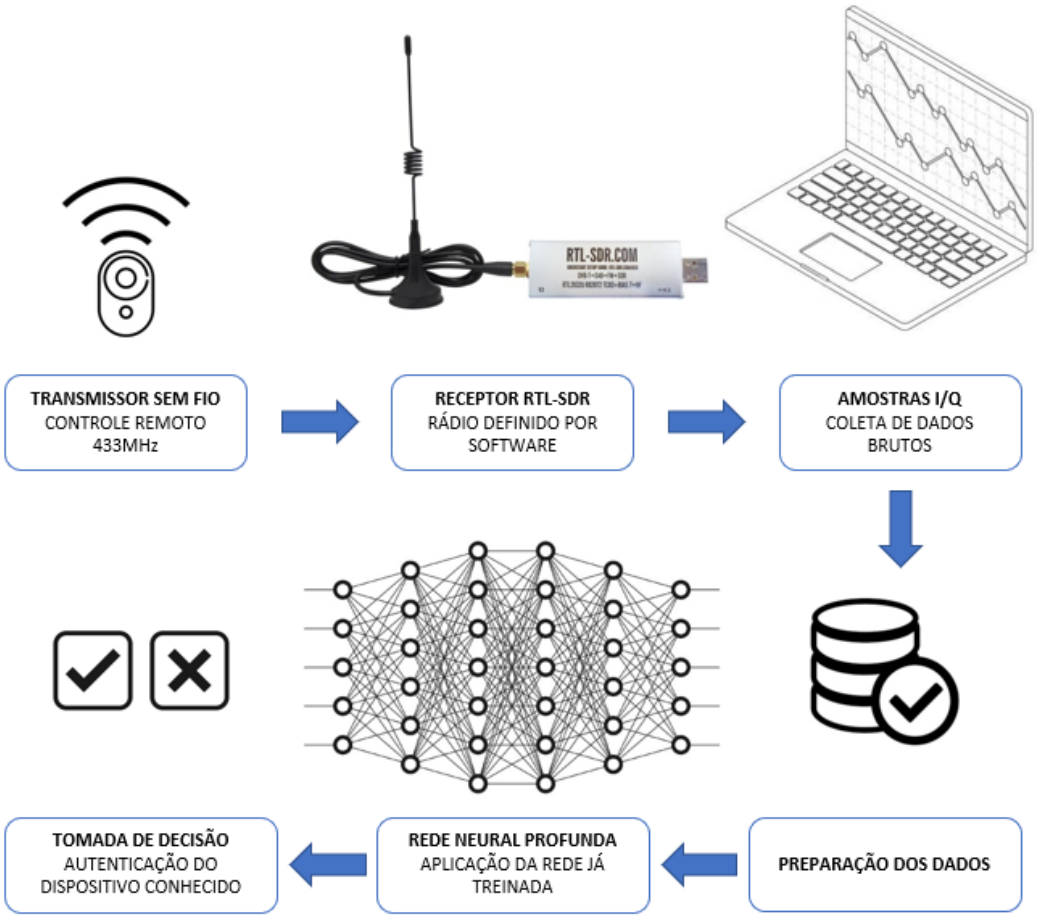
Figura 5: Treinamento da rede neural



Fonte: o autor.

Uma vez que a rede tenha sido treinada, ela pode ser aplicada de acordo com a descrição apresentada na Figura 6. O esquema de utilização envolve receber os sinais dos controles emissores através do rádio definido por *software*, processar esses sinais, preparar os dados e, por fim, aplicar a rede já treinada. Dessa forma, espera-se que a rede seja capaz de identificar a qual dos quatro emissores o sinal pertence.

Figura 6: Utilização da rede neural



Fonte: o autor.

3.1 TRANSMISSORES SEM FIO

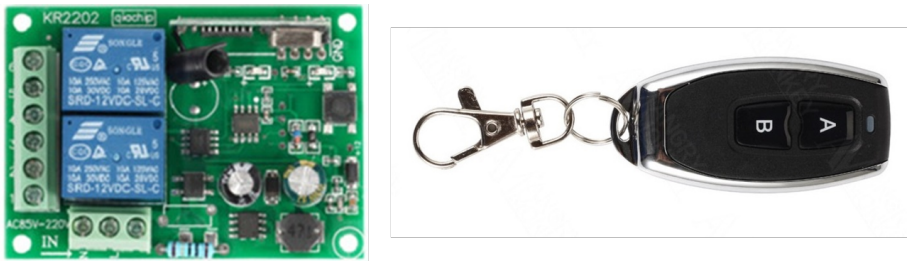
As faixas de frequência *Industrial Sientific and Medical* (ISM) são bandas reservadas internacionalmente para o desenvolvimento industrial, científico e médico. Tratam-se de partes do espectro de frequência para desenvolvimentos livres, sem a necessidade de licenciamento, desde que seguidas as normas de limitação de potência de transmissão e de técnicas de modulação para essas faixas (TELECO, 2022).

Este padrão foi internacionalmente difundido e adotado em diversos países. No Brasil, foi adotado com algumas ressalvas. A faixa de 433 MHz a 435 MHz, por exemplo, não foi incluída como ISM, no entanto sua utilização pode ser feita desde que utilizados equipamentos de radiação restrita, em áreas internas de edificações e com potência irradiada limitada ao valor máximo de 10 mW (ANATEL, 2004).

Sendo assim, para o presente estudo, foram adquiridos módulos de rádio, de curto alcance, de baixo custo e de fabricação chinesa, com frequência central de 433,92 MHz. O primeiro trata-se de um módulo relé, Figura 7, da fabricante QIACHIP

(*https://qiachip.com*). O módulo é composto por um receptor e um controle transmissor que liga e desliga os relés remotamente.

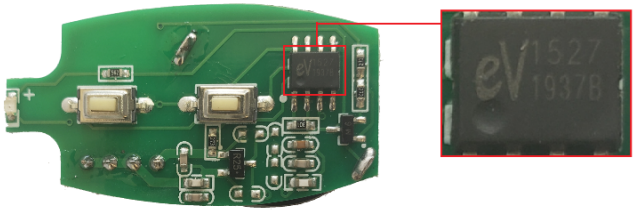
Figura 7: Módulo receptor e controle 433 MHz



Fonte: o autor.

O controle tem como núcleo um codificador *One-Time Pad* (OTP) EV1527, conforme pode ser visto na Figura 8.

Figura 8: Circuito do transmissor com codificador OTP EV1527

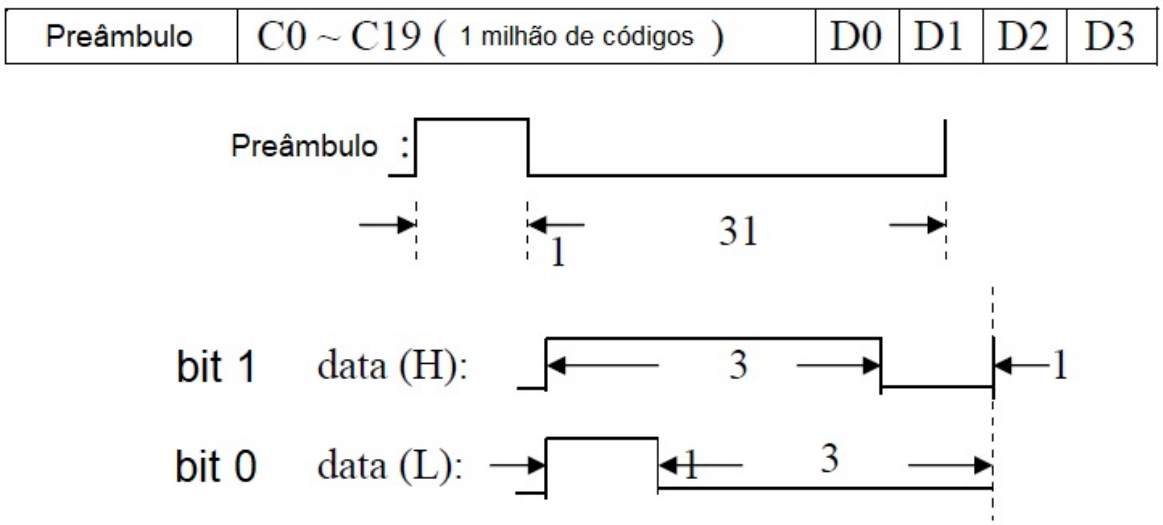


Fonte: o autor.

Basicamente, de acordo com o *datasheet*, quando o botão é pressionado, o controle emite um sinal de radiofrequência no qual a mesma sequência binária é transmitida repetidamente, com um pequeno intervalo entre as repetições, chamado preâmbulo. Cada sequência consiste em dois tipos de ondas quadradas: um pulso largo, com uma breve interrupção no final, representando o bit 1 e um pulso curto, com uma pausa, representando o bit 0. O sinal é composto por um preâmbulo e na sequência 24 bits, dos quais os 20 primeiros bits representam o código que individualiza o controle e os 4 últimos bits representam o comando ou o botão pressionado. Com 20 bits para cada controle temos mais de 1 milhão de possíveis controles (2^{20}), dificultando bastante que um controle acione um receptor para o qual ele não foi pareado (SILVAN, 2022).

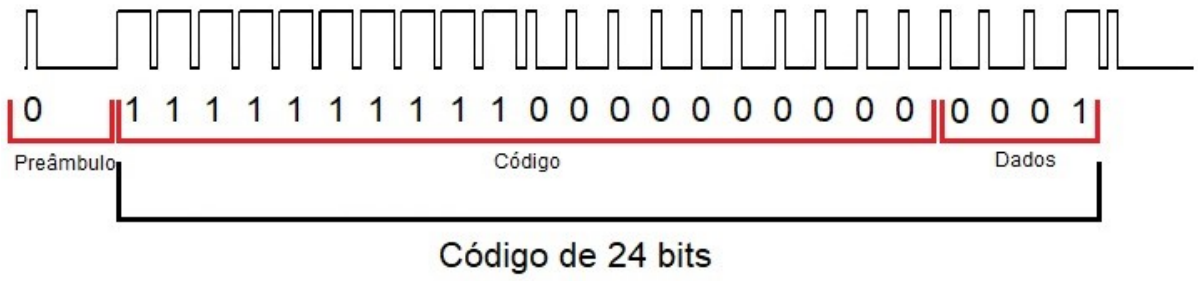
Considere que se deseja transmitir uma sequência binária com o código do controle composto pelos dígitos: 11111111110000000000 e o comando/botão: 0001. Nesse caso, o sinal emitido pelo controle teria a forma apresentada na Figura 10.

Figura 9: Formato de saída do sinal do controle



Fonte: Adaptado de (SILVAN, 2022, p.2).

Figura 10: Sinal de saída do controle para o código 111111111100000000000001



Fonte: O autor.

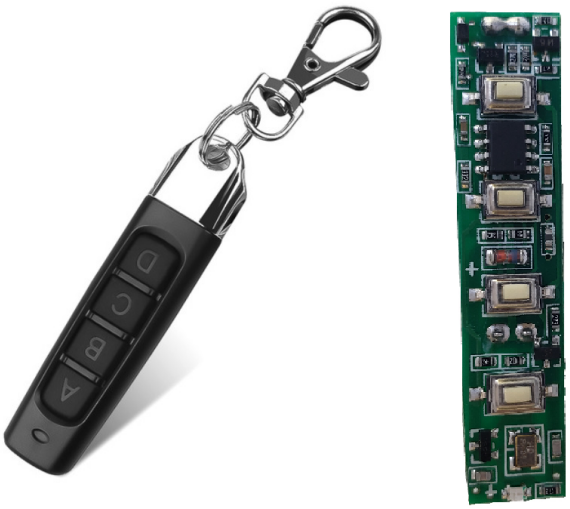
Os vinte primeiros bits representam o código do controle. Portanto, controles diferentes possuirão códigos diferentes e emitirão sinais diferentes, mesmo pareados com o mesmo receptor, pois um receptor pode parear-se a mais de um controle.

Tal situação foge aos interesses do estudo, pois devemos fornecer à rede neural sinais semelhantes para que possam ser diferenciados com base apenas nas características únicas impressas no sinal pelo *hardware* emissor. Os transmissores devem emitir a exata mesma sequência de bits.

Para contornar essa situação foram adquiridos três controles copiadores, que clonam os 24 bits do sinal do controle original. Os controles copiadores adquiridos são da marca KEBIDU e um exemplar pode ser visto na Figura 11.

Infelizmente não foi possível conseguir maiores informações acerca dos controles copiadores adquiridos, tais como *datasheet*, fabricante ou o circuito integrado codifi-

Figura 11: Controle copiador KEBIDU e seu circuito interno



Fonte: O autor.

cador que, como podemos ver pela Figura 11, não possui descrição impressa.

O próximo passo é configurar o rádio definido por *software* para coletar os sinais emitidos pelos transmissores.

3.2 RECEPTOR

RTL-SDR é um *dongle* de baixo custo, que pode ser usado como um rádio definido por *software*, capaz de receber sinais de rádio na faixa de 500 kHz até 1,75 GHz, sem uso de internet.

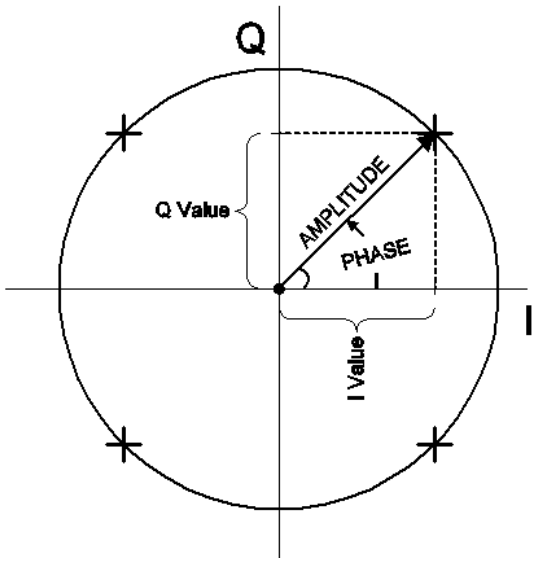
Figura 12: *Dongle* RTL-SDR 2832U

Fonte: Adaptado de (RTL-SDR.COM, 2019).

Tal dispositivo é capaz de coletar o sinal na forma de dados complexos I/Q,

em que I representa a fase, expressa por um número real, e Q representa a quadratura, expressa por um número imaginário. Podemos observar na Figura 13 a relação entre a amplitude relativa do sinal e as amostras I/Q. Com base nessa observação chegamos à Equação 9.

Figura 13: Relação I/Q e Amplitude



Fonte: O autor.

$$Amplitude = \sqrt{I^2 + Q^2} \tag{9}$$

3.3 COLETA DE DADOS

Todo o código-fonte utilizado neste estudo foi escrito na linguagem de programação Python, versão 3.10.10, e pode ser encontrado no repositório do autor (GONÇALVES, 2023). Primeiramente foi definida uma taxa de amostragem de dados de 2,4 MHz. A frequência central foi definida em 433,9 MHz. Dessa forma, o RTL-SDR captura sinais com frequências entre 432,7 MHz e 435,1 MHz. O ganho foi definido em 4 dB para não saturar o sinal. A interface Pyhton para o SDR exige que o número de amostras seja múltiplo de 256, por isso foram coletadas um pouco mais de dezesseis milhões de amostras por radio transmissor ($256 * 65536 = 16777216$).

O transmissor foi mantido a cerca de meio metro da antena. Ao pressionar o botão para transmissão do sinal, era então executado o *script* para iniciar a captura através do SDR. Assim, repetiu-se o processo, coletando as amostras para cada um dos quatro transmissores. O próximo passo é o processamento desses sinais.

3.3.1 Processamento do Sinal

Técnicas de processamento de sinal melhoram os processos subsequentes de análise e conferem maior legibilidade. As técnicas utilizadas nesse estudo foram a aplicação de um filtro Butterworth e a normalização do sinal.

O filtro Butterworth permite alcançar uma resposta em frequência plana na faixa de passagem, enquanto atenua as frequências fora da faixa desejada. Com isso, é possível atenuar, por exemplo, os ruídos. Para o estudo, foi utilizado um filtro do tipo passa-baixa de ordem três.

A normalização é um passo essencial para padronizar a faixa de amplitude dos sinais. Durante o processo de normalização, o sinal é ajustado para uma escala pré-definida entre 0 e 1. Dessa forma, em cada conjunto amostrado por transmissor, o menor valor de amplitude se torna zero, o maior se torna um e os demais valores são ajustados proporcionalmente entre esses extremos. Esse processo é particularmente útil para que a rede não aprenda a diferenciar os transmissores com base apenas na amplitude do sinal. A normalização ajuda a garantir que os sinais possam ser comparados, analisados e processados de forma consistente, independentemente de suas amplitudes originais, facilitando a interpretação. Em seguida é necessário preparar esses dados para serem usados na rede.

3.4 BASE DE DADOS

Com os sinais coletados é preciso organizá-los. Inicialmente tínhamos as componentes reais e imaginárias do sinal representadas em *arrays*. Com uso da Equação 9 passamos a ter uma única *array* de amplitude, a qual foi filtrada e normalizada. Para esses dados serem usados no treinamento e no teste da rede, remodelamos essa *array* de coluna única em uma de formato 32x32, bidimensional. Em seguida, separamos 75% das amostras para serem usadas para treino e os 25% restantes para serem usadas para teste ou avaliação da rede depois de treinada.

Outra etapa importante é a rotulação desses dados. Por se tratar de aprendizado de máquina supervisionado, é necessário rotular as amostras para que a rede possa associar e classificar a qual transmissor esses dados pertencem. Como temos quatro transmissores, a saída da rede terá quatro neurônios correspondentes. Consequentemente, os dados de rotulação devem ser no formato de *arrays* de 4 colunas para coincidir com a saída da rede. Por exemplo, para os dados de treino do primeiro transmissor criamos uma *array* de rótulos composta por quatro colunas, sendo a primeira coluna formada por números 1 e as demais por números 0. Dessa forma, a rede sempre será informada que esses dados devem acionar a saída do primeiro neurônio, que representa o primeiro transmissor. Processo semelhante é adotado para a criação dos demais rótulos, modificando-se apenas qual das quatro colunas será composta por números 1, com o número dessa coluna

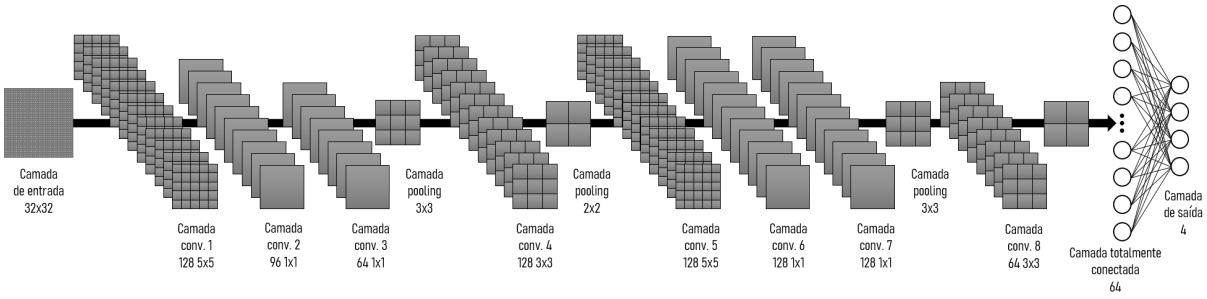
correspondendo ao número do transmissor de origem dos dados.

Por fim, salvamos a base de dados para que, em outro *script*, possamos treinar e avaliar a rede. Mas antes disso precisamos definir a arquitetura da rede.

3.5 ARQUITETURA E TREINAMENTO DA CNN

A arquitetura escolhida para a rede pode ser vista na Figura 14. Em um primeiro momento foi utilizada a arquitetura proposta por (PASHAM, 2019), no entanto foram necessários ajustes para melhor aprendizado. Foram incluídas mais camadas convolucionais e substituídas a maioria das funções de ativação para a PReLU, assemelhando-se a arquiteturas voltadas para reconhecimento de imagens.

Figura 14: Arquitetura da CNN implementada



Fonte: O autor.

A rede possui como entrada uma matriz bidimensional de tamanho 32x32. Em seguida, é composta por camadas convolucionais, sendo a primeira composta por 128 filtros convolucionais de tamanho 5 com escorregamento ou *stride* 1. A segunda camada convolucional possui 96 filtros, tamanho 1 e *stride* 1. A terceira camada possui 64 filtros, tamanho 1 e *stride* 1. As três utilizam o *zero padding* e a função de ativação PReLU, conforme Equação 2.

Na sequência temos uma camada de *pooling* com a função máximo e *stride* 2. A quarta camada convolucional é composta por 128 filtros convolucionais de tamanho 3, *stride* 1 e função de ativação ReLU, conforme Equação 1. Essa camada utiliza a opção para decaimento de peso ou regularização de peso L2, para melhor ajuste do modelo. Temos ainda a segunda camada de *pooling* com a função Máximo e *stride* 2.

Em seguida, temos mais três camadas convolucionais usando função de ativação PReLU, *zero padding* e *stride* 1. Dessas, a quinta camada convolucional possui 128 filtros de tamanho 5. A sexta e a sétima possuem 128 filtros de tamanho 1 cada. Adiante temos a terceira camada de *pooling*, dessa vez com a função Média e *stride* 2.

Temos ainda a oitava camada convolucional com 64 filtros, tamanho 3, ativação ReLU, regularização de peso L2, e *stride* 1. Nossa útlima camada de *pooling* vem com a função máximo e *stride* 2. Por fim, temos uma camada totalmente conectada com 64

neurônios e função de ativação Tangente Hiperbólica ligados à camada de saída com 4 neurônios com função de ativação compartilhada Softmax.

3.5.1 Treinamento da rede

Para o treinamento da rede foi utilizado o Python 3.10.10 em ambiente Windows 10 com a biblioteca TensorFlow versão 2.10.0. Além disso, para acelerar o treinamento da rede, foi utilizada a capacidade computacional da placa de vídeo. As placas de vídeo da marca NVIDIA possuem uma plataforma de computação paralela chamada CUDA, que permite utilizar o alto potencial de processamento paralelo proporcionado por uma placa de vídeo. Para isso foram utilizados os *softwares* CUDA Toolkit na versão 11.2 e CuDNN na versão 8.1.

Em termos de *hardware* foi utilizada a placa de vídeo da NVIDIA RTX 3060, modelo com 12GB de memória de vídeo - VRAM. A placa possui 3584 Cores CUDA e 360 GB/s de largura de banda da memória.

4 RESULTADOS E DISCUSSÕES

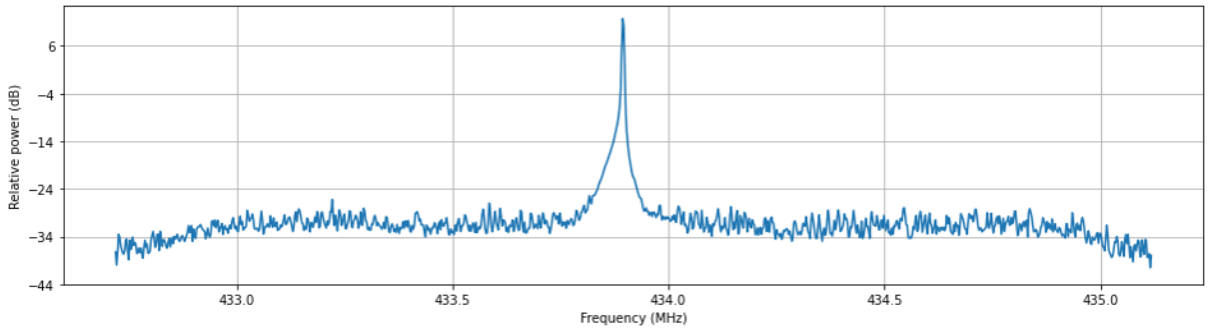
Nesta Seção, apresentaremos os resultados dos experimentos conduzidos, desde a coleta dos dados, treinamento da rede, até a aplicação da rede já treinada. Adicionalmente, serão discutidos aspectos relacionados ao desempenho da rede, como a influência de hiperparâmetros, arquitetura da rede e estratégias de treinamento. Por meio dessa análise crítica, será possível compreender a efetividade da rede neural na identificação dos dispositivos emissores, destacando suas limitações e possíveis melhorias para aplicações futuras.

4.1 COLETA E ANÁLISE DO SINAL

O *script* em Python utilizado para a coleta dos dados e processamento dos sinais, de nome data-collection-&-signal-processing, está disponível no repositório GitHub do autor (GONÇALVES, 2023). Primeiramente faremos a análise dos sinais comparando o controle original com apenas um dos controles copiadores, tendo em vista que os três controles copiadores utilizados apresentaram características semelhantes.

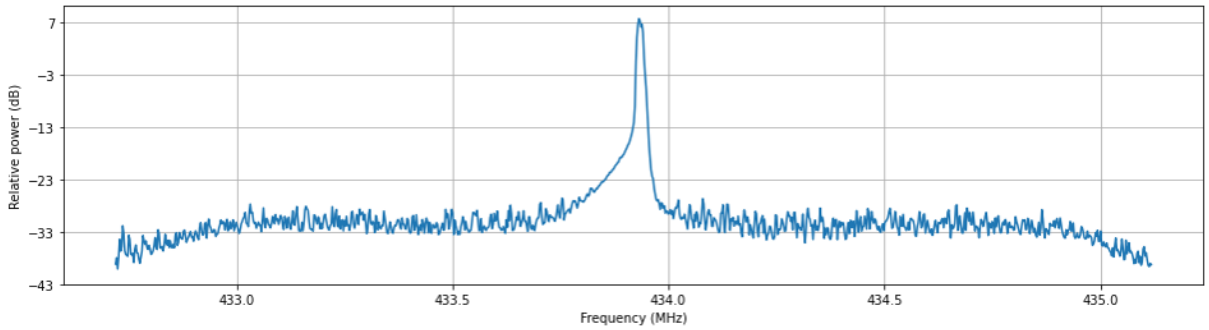
Feita a coleta dos dados na forma I/Q podemos aplicar uma Transformada de Fourier Rápida e plotar os dados no domínio da frequência. Como podemos observar pelas Figuras 15 e 16, ambos os controles apresentam frequências próximas à frequência central de 433,9 MHz.

Figura 15: Sinal do controle original no domínio da frequência



Fonte: O autor.

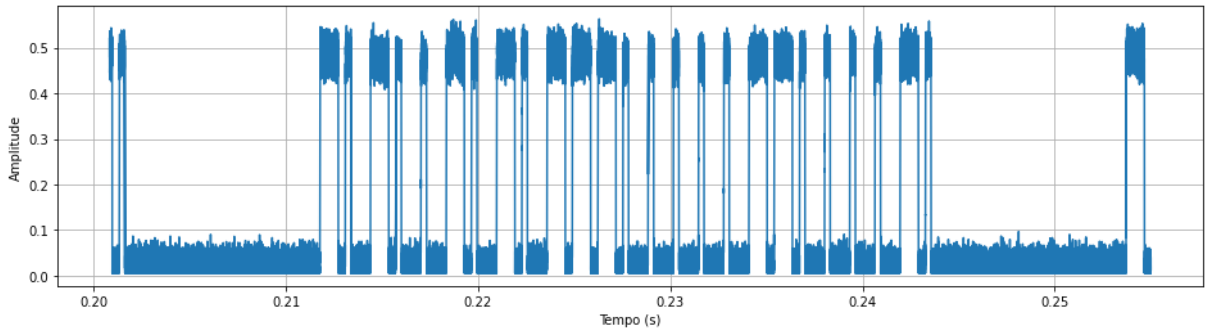
Figura 16: Sinal do controle copiador no domínio da frequência



Fonte: O autor.

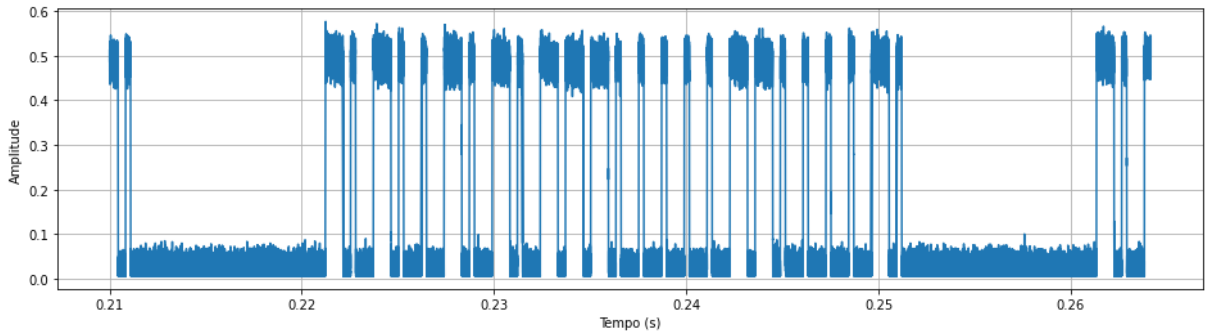
É importante analisar também o sinal no domínio do tempo. Implementando a equação da amplitude (9) e definindo uma lista t com valores para o eixo de tempo, podemos plotar nosso sinal no domínio do tempo.

Figura 17: Sinal do controle original no domínio do tempo



Fonte: O autor.

Figura 18: Sinal do controle copiador no domínio do tempo

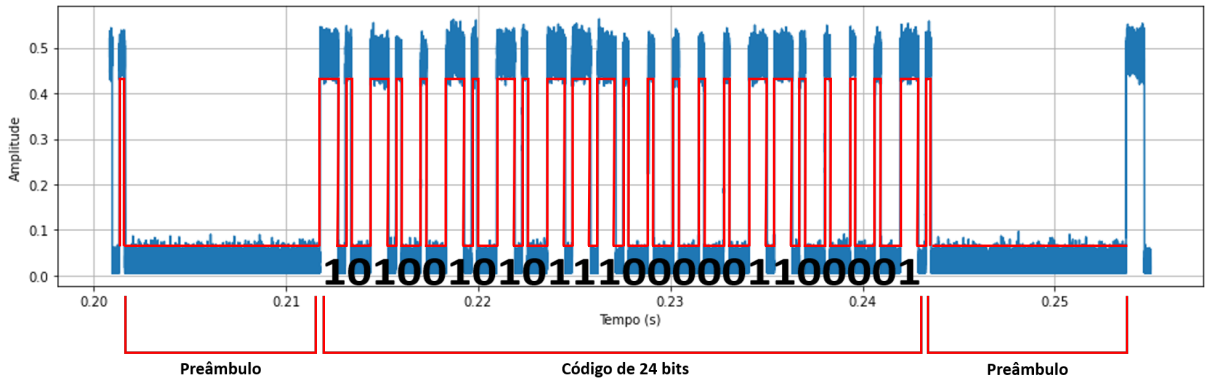


Fonte: O autor.

Com base na Figura 9, podemos analisar as Figuras 17 e 18 e chegar à conclusão de que ambos os controles, original e copiador, transmitem exatamente a mesma sequência

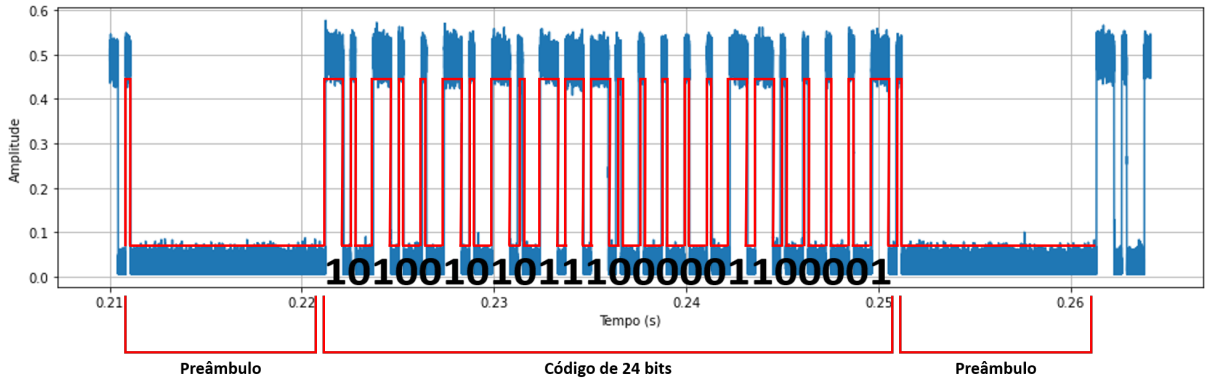
de 24 bits, 101001010111000001100001, em que os primeiros 20 bits representam o código do controle e os 4 últimos representam o botão acionado, que, nesse caso, foi o botão 0001. Dessa forma, com a análise apresentada nas Figuras 19 e 20, podemos perceber que os controles atendem aos critérios propostos na pesquisa.

Figura 19: Análise do sinal do controle original



Fonte: O autor.

Figura 20: Análise do sinal do controle copiador



Fonte: O autor.

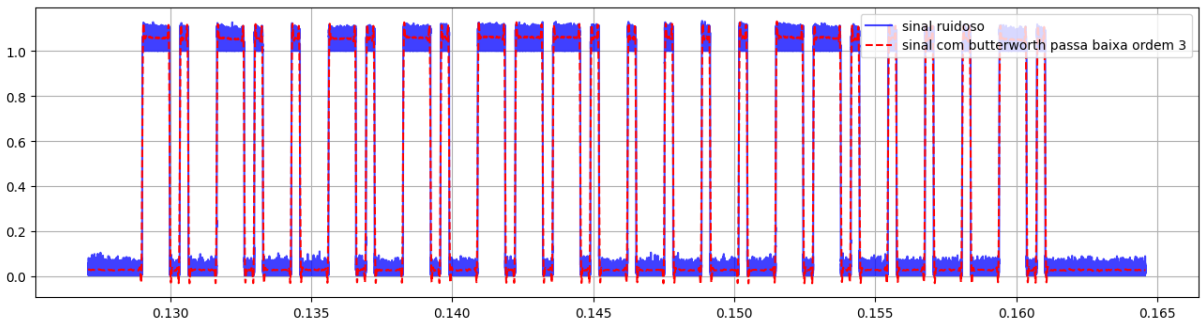
O próximo passo é o tratamento desses sinais.

4.1.1 Aplicação do Filtro Digital

Através das Figuras 17 e 18 é possível perceber ruído excessivo presente no sinal. Nos testes realizados, esse excesso causou baixo desempenho no treinamento e também impediu a rede de generalizar para novos dados, fazendo com que ela obtivesse baixa acurácia. É possível que o ruído interfira nos padrões e informações relevantes presentes nos sinais, prejudicando a capacidade da rede neural em identificar e extrair características significativas.

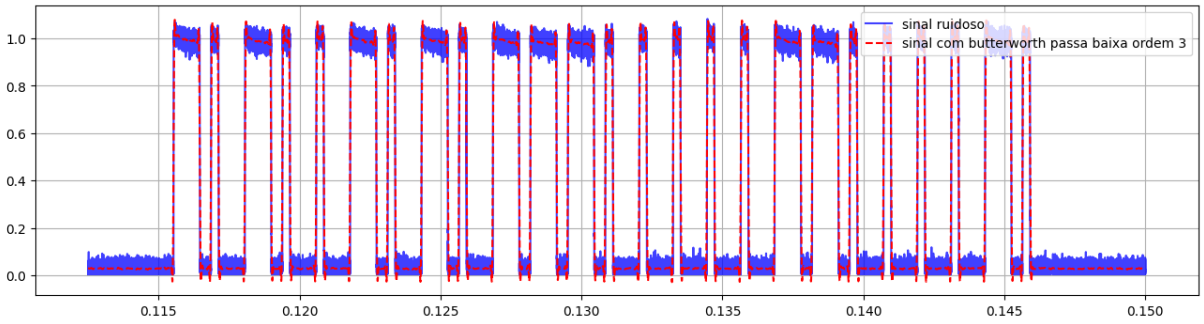
Por outro lado, esse trabalho parte da premissa de que os sinais de radiofrequência possuem atributos físicos intrínsecos distintos RF-DNA, os quais se confundem com o próprio ruído. Para encontrar um filtro que atenuasse o ruído sem prejudicar o RF-DNA utilizou-se o método da tentativa e erro. O resultado pode ser visto nas figuras 21 e 22.

Figura 21: Comparativo ruído x filtro do sinal do controle original



Fonte: O autor.

Figura 22: Comparativo ruído x filtro do sinal do controle copiador



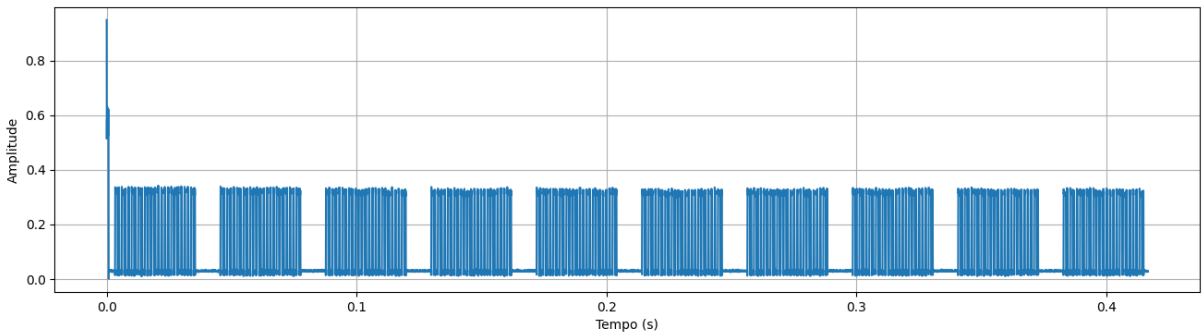
Fonte: O autor.

4.1.2 Normalização do Sinal

Outra etapa importante diz respeito à normalização do sinal. Sem esse tratamento a rede aprendeu a diferenciar os sinais dos transmissores com base somente na amplitude. Isso a torna inútil para aplicações reais, onde diversos fatores interferem na intensidade do sinal captado, tais como distância entre emissor e receptor, potência do *hardware* transmissor, dentre outros.

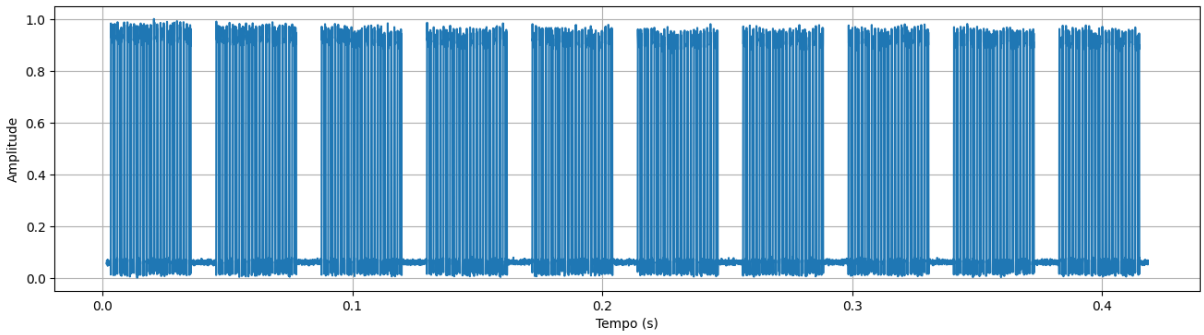
As Figuras 23 e 24 representam uma janela de tempo maior de captura dos sinais, onde é possível observar os resultados da filtragem e normalização aplicados ao controle original. Na Figura 23 temos um pico de amplitude no começo, seguido de vários blocos de 24 bits com amplitude menor. Todos os controles transmissores apresentaram

Figura 23: Sinal do controle original não normalizado



Fonte: O autor.

Figura 24: Sinal do controle original filtrado e normalizado entre 0 e 1



Fonte: O autor.

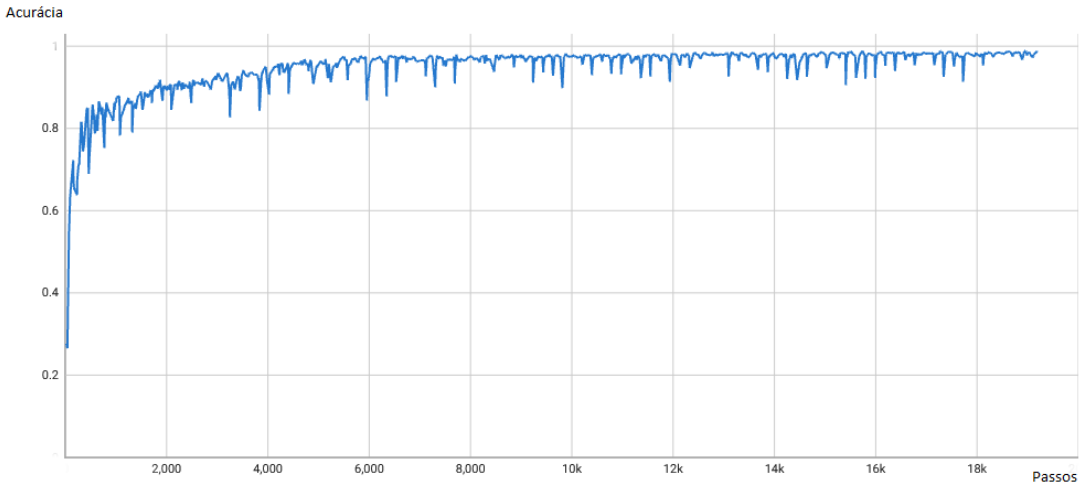
comportamento semelhante, variando somente a amplitude do pico inicial e a dos blocos. Na normalização, o pico inicial foi descartado e no restante do sinal foi atribuído o valor 1 ao pico de maior amplitude e o valor 0 ao vale de menor amplitude, ajustando os valores intermediários proporcionalmente, conforme se vê na Figura 24.

4.2 TREINAMENTO E VALIDAÇÃO DA REDE

As bases de dados de treinamento e de validação, bem como o *script* em Python utilizado, de nome convnet-radio-identification, estão disponíveis no repositório GitHub do autor (GONÇALVES, 2023). Inicialmente foram aplicadas as arquiteturas de redes propostas por (PASHAM, 2019) e (RIYAZ et al., 2018), as quais não se mostraram eficazes para as bases de dados desse estudo. Através do método de tentativa e erro e utilizando-se de técnicas de redes voltadas para o reconhecimento de imagens chegou-se na arquitetura proposta nesse trabalho, conforme visto na Figura 14.

A Figura 25 mostra a acurácia da rede implementada. O eixo y (vertical) do gráfico representa a porcentagem de acertos da rede, enquanto o eixo x (horizontal) indica o número de passos.

Figura 25: Acurácia da rede no treinamento

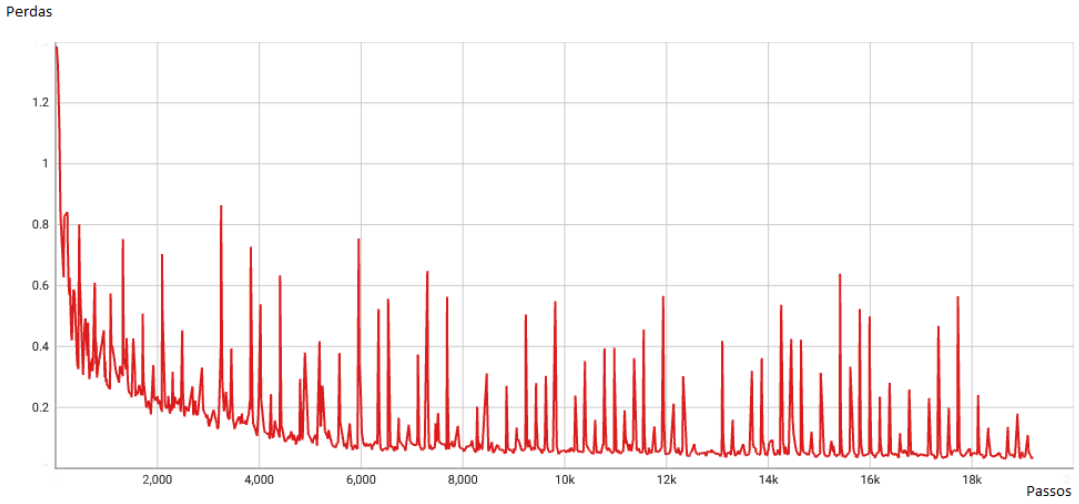


Fonte: O autor.

Ao observar a acurácia ao longo do tempo é possível perceber um aumento contínuo na precisão da rede até a estabilização em torno de 98%, quando atinge 19200 passos de treinamento. Além desse ponto não há acréscimos significativos.

Outra questão a ser observada são as oscilações na curva. Acredita-se que a presença do ruído ou perturbações aleatórias no conjunto de dados pode levar a oscilações na acurácia. O ruído pode introduzir incerteza na classificação e fazer com que a acurácia flutue ao longo do tempo, mesmo que o desempenho geral seja bom. No entanto, sempre que se tentava atenuar mais o ruído por meio do processamento do sinal, o RF-DNA também ficava prejudicado impedindo a rede de aprender.

Figura 26: Perdas da rede no treinamento



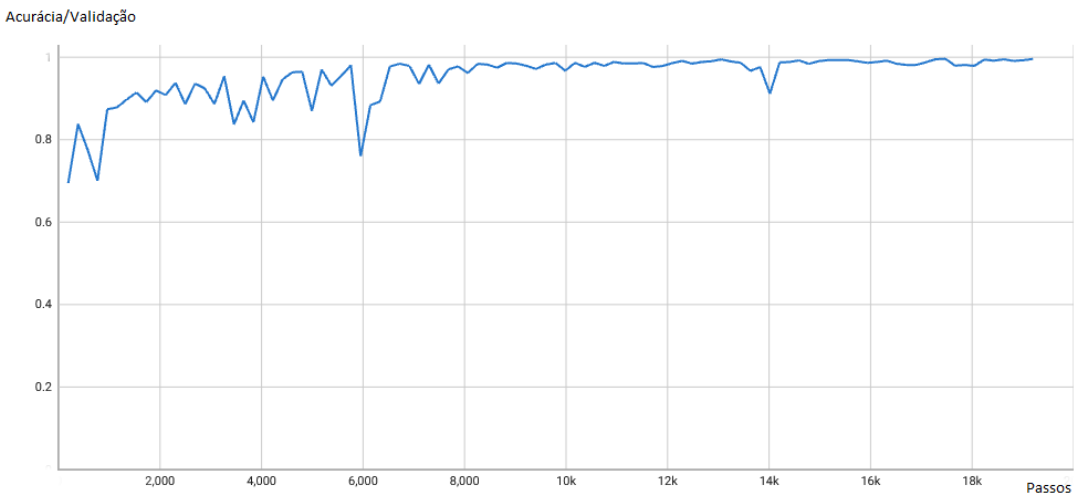
Fonte: O autor.

O gráfico de perda (Figura 26) quantifica o quão distante as previsões da rede estão dos valores reais dos dados de treinamento. À medida que o treinamento progride espera-se que a rede busque minimizar a função de perda, ajustando os pesos e os parâmetros para que as previsões se aproximem o máximo possível dos valores reais. O eixo y (vertical) do gráfico representa o valor da perda, enquanto o eixo x (horizontal) representa o número de passos do treinamento.

Podemos observar que a perda está diminuindo consistentemente, indicando que a rede está aprendendo a partir dos dados de treinamento. Uma possível causa das oscilações presentes na curva pode ser relacionada a problemas de otimização dos parâmetros da rede, como por exemplo uma taxa de aprendizagem alta, que faz com que a rede oscile em torno de um mínimo local em vez de convergir suavemente. No entanto, por meio de tentativa e erro não se chegou a parâmetros melhores do que aqueles implementados.

Conforme mencionado na Seção 3.4, foram reservados 25% dos dados coletados para a validação da rede. Esse conjunto de validação é usado para verificar se a rede está generalizando bem e não está apenas memorizando os exemplos de treinamento. Quando a acurácia no conjunto de treinamento está aumentando, mas a acurácia no conjunto de validação está diminuindo ou estagnada, pode ser um sinal de que a rede está sofrendo de *overfitting* (sobreajuste), ou seja, está se adaptando demais aos dados de treinamento específicos e não generalizando bem para novos dados. No presente estudo o sobreajuste ocorreu quando se aumentou além do proposto na Figura 14 a quantidade de neurônios e de camadas convolucionais.

Figura 27: Acurácia da rede na validação

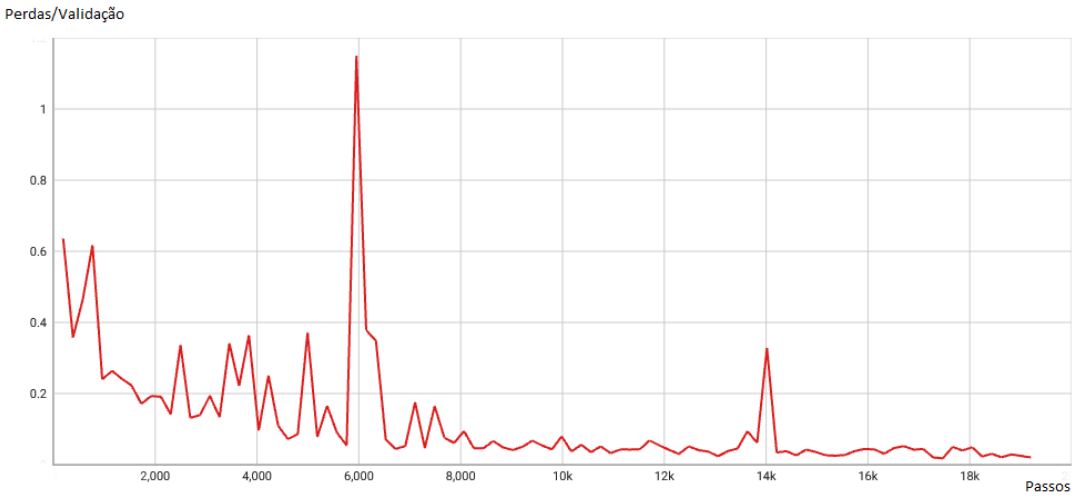


Fonte: O autor.

A Figura 27 exibe a taxa de acurácia da rede neural no conjunto de validação. Podemos perceber resultados na faixa de 99% de acurácia, sem oscilações significativas.

A Figura 28 mostra como a perda da rede evoluiu no conjunto de validação ao longo do tempo. É possível observar que, a partir de certo ponto, a perda diminui de forma consistente, indicando que a rede está aprendendo padrões gerais dos dados e é capaz de fazer previsões precisas em exemplos não vistos anteriormente.

Figura 28: Perdas da rede na validação



Fonte: O autor.

4.3 APLICAÇÃO DA REDE TREINADA

O modelo treinado, bem como o *script* em Python utilizado para aplicação da rede, de nome convnet-application, está disponível no repositório GitHub do autor (GONÇALVES, 2023). Com o intuito de experimentar a rede em uma aplicação prática, desenvolveu-se um *script* que verifica, através do equipamento de rádio definido por *software*, a emissão de sinais na faixa de 433,9 MHz. Uma vez detectado, o *script* coleta os dados durante um certo intervalo e aplica o processamento de sinais. Por fim, os dados são submetidos à análise da rede treinada, a qual exibe a avaliação em forma de um número correspondente ao controle transmissor que a rede julga ter emitido o sinal. Tal esquema está representado na Figura 6.

A rede neural inicialmente obteve resultados promissores durante o processo de treinamento, demonstrando uma alta taxa de acerto e um bom desempenho em um conjunto de dados de treinamento bem selecionado. No entanto, ao ser implementada em uma solução prática, a rede neural enfrentou desafios de generalização devido à diferença entre os dados coletados em tempo real e os dados utilizados no treinamento. Essa discrepância entre os dados de treinamento e os dados reais afetou negativamente a taxa de acerto da rede neural, que em testes práticos ficou em torno de 80% de acertos. Essa discrepância pode ser atribuída a diversas razões, como variações na coleta e tratamento

dos dados em tempo real ou até mesmo a presença de ruídos não considerados anteriormente. Deixaremos para um trabalho futuro implementar estratégias de adaptação e aprimoramento para lidar com a coleta de dados em tempo real.

5 CONCLUSÃO

Ao longo desse trabalho, exploramos o potencial e a aplicação das redes neurais convolucionais (CNNs) na área de reconhecimento de sinais. Foram desenvolvidos métodos para amostragem de sinais, processamento dos dados, construção de base de dados, implementação e utilização de uma rede neural convolucional para autenticação de dispositivos de transmissão.

Podemos concluir que a rede neural convolucional proposta demonstra a viabilidade de um sistema de autenticação que identifique um transmissor através de suas ondas de rádio frequência. Podemos afirmar ainda que é viável a aplicação de tal sistema como uma camada adicional de segurança em redes de dados contendo dispositivos sem fio de baixo custo.

Contudo, a tecnologia aqui proposta implica em treinar novamente a rede neural toda vez que um novo dispositivo for removido ou adicionado à rede de dados. Uma possível forma de contornar esse problema é utilizando os serviços de treinamento de rede em nuvem, disponíveis atualmente.

Com 98% de acurácia no treinamento e 99% na validação, a CNN proposta se mostrou capaz de generalizar para além dos dados de treinamento. Porém, na aplicação prática, o desempenho não foi satisfatório. Faz-se necessário aprimorar o sistema de coleta de dados em tempo real para aproximar os resultados práticos aos do treinamento.

A performance alcançada reforça a hipótese de que os dispositivos eletrônicos de transmissão sem fio imprimem variações físicas únicas nos sinais que emitem e que essas variações podem ser utilizadas como uma assinatura ou "impressão digital" do dispositivo emissor. Apesar dos desafios, as CNNs provaram ser uma ferramenta poderosa e promissora, com capacidade de identificar esses padrões únicos e diferenciar inequivocamente um dispositivo de outro.

Trabalhos futuros poderão melhorar a precisão da rede em aplicações práticas e avaliar melhor o impacto de algumas adversidades sobre a precisão da rede, tais como: distância entre transmissor e receptor, relação sinal ruído - SNR, desgaste natural do equipamento transmissor, tipo de modulação do sinal, dentre outras.

REFERÊNCIAS

ALVES, G. *Entendendo Redes Convolucionais (CNNs)*. 2018. Disponível em: <https://medium.com/neuronio-br/entendendo-redes-convolucionais-cnns-d10359f21184>. Acesso em: Novembro 2019.

ANATEL. Anexo À resoluÇÃo no 365, de 10 de maio de 2004. *Diário Oficial da União*, Brasília, DF, 2004. Disponível em: <https://www.anatel.gov.br/Portal/verificaDocumentos/documento.asp?null&filtro=1&documentoPath=biblioteca/resolucao/2004/Anexo_res_365_2004.pdf>.

BRADLEY, J.; BARBIER, J.; HANDLER, D. Embracing the internet of everything to capture your share of $14.4 trillion. Cisco Systems Inc., 2013. Disponível em: <https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoE_Economy.pdf>.

BRASIL. Decreto nº 9.854, de 25 de junho de 2019. *Diário Oficial da União*, Brasília, DF, 2019. Disponível em: <https://www.planalto.gov.br/ccivil_03/_ato2019-2022/2019/decreto/d9854.htm>.

CASS, S. A $40 software-defined radio. IEEE Spectrum, 2013. Disponível em: <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6545114>.

CASTRO, L. N. d.; ZUBEN, F. J. V. *Redes Neurais Artificiais*. [S.l.]: DCA/FEEC/Unicamp, 2003. Disponível em: <ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia006_03/topico5_03.pdf>. Acesso em: Novembro 2019.

CISCO. The internet of everything. global public sector economic analysis. Cisco Systems Inc., 2013. Disponível em: <https://www.cisco.com/c/dam/en_us/about/business-insights/docs/ioe-value-at-stake-public-sector-analysis-faq.pdf>.

CUI, L. et al. A survey on application of machine learning for internet of things. *International Journal of Machine Learning and Cybernetics*, v. 9, p. 1399–1417, 6 2018.

EVANS, D. The internet of everything. how more relevant and valuable connections will change the world. Cisco Systems Inc., 2012. Disponível em: <https://www.cisco.com/c/dam/global/en_my/assets/ciscoinnovate/pdfs/IoE.pdf>.

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

GONÇALVES, K. J. *Repositório GitHub*. [S.l.]: GitHub, 2023. <https://github.com/eukaio/fingerprinting_radios_w_ML>.

HE, K. et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. Microsoft, 2015. Disponível em: <https://arxiv.org/pdf/1502.01852v1.pdf>.

JARA, A. J.; LADID, L.; GÓMEZ-SKARMETA, A. F. The internet of everything through ipv6: An analysis of challenges, solutions and opportunities. *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.*, v. 4, p. 97–118, 2013.

KEOH, S. L.; KUMAR, S. S.; TSCHOFENIG, H. Securing the internet of things: A standardization perspective. *IEEE INTERNET OF THINGS JOURNAL*, v. 1, n. 3, p. 265–275, 6 2014.

LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, v. 86, n. 11, p. 2278–2324, 1998.

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: Novembro 2019.

LEO, M. et al. A federated architecture approach for internet of things security. *2014 Euro Med Telco Conference - From Network Infrastructures to Network Fabric: Revolution at the Edges, EMTC 2014*, 11 2014.

MCCULLOCH, W. S. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, v. 5, p. 115–133, 1943.

PASHAM, N. *Authenticating 'low-end wireless sensors' with deep learning + SDR*. 2019. Disponível em: <https://nihal-pasham.medium.com/authenticating-low-end-wireless-sensors-with-deep-learning-sdr-f059ed6d8840>. Acesso em: Novembro 2019.

PATEL, H. J.; TEMPLE, M. A.; BALDWIN, R. O. Improving zigbee device network authentication using ensemble decision tree classifiers with radio frequency distinct native attribute fingerprinting. *IEEE Transactions on Reliability*, v. 64, n. 1, p. 221–233, 2015.

RIYAZ, S. et al. Deep learning convolutional neural networks for radio identification. *IEEE Communications Magazine*, v. 56, n. 9, p. 146–152, 2018.

RTL-SDR.COM. *About RTL-SDR*. 2019. Disponível em: <https://www.rtl-sdr.com/about-rtl-sdr/>. Acesso em: Novembro 2019.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning internal representations by error propagation. University of California. Institute for Cognitive Science, 1985.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, v. 3, n. 3, p. 210–229, 1959.

SILVA, W. S. et al. *Introdução a Rádios Definidos por Software com aplicações em GNU Radio*. 2015. Disponível em: <http://sbrc2015.ufes.br/wp-content/uploads/Ch5.pdf>. Acesso em: Agosto 2019.

SILVAN. Datasheet: Ev1527 otp encoder. Silvan Chip Electronics Tech.Co.,Ltd., 2022. Disponível em: <http://www.sc-tech.cn/en/EV1527.pdf>.

TEIXEIRA, F. A. et al. Defending internet of things against exploits. *IEEE Latin America Transactions*, v. 13, n. 4, p. 1112–1119, 2015.

TELECO. *Redes Wi-Fi I: Espectro de Frequência ISM*. 2022. Disponível em: <https://www.teleco.com.br/tutoriais/tutorialredeswifi1/pagina_5.asp>. Acesso em: Abril 2022.

WANG, X. et al. Identification and authentication for wireless transmission security based on rf-dna fingerprint. *EURASIP Journal on Wireless Communications and Networking*, v. 2019, p. 1687–1499, 2019.