Estudo de aplicação de *autoencoders* em sistemas de comunicação digital usando rádio definido por *software*

Thamer R. Leme, Isaque H. Gonçalves, Evandro C. Vilas Boas, Felipe Augusto P. de Figueiredo
Laboratório de Cyber Segurança e Internet das Coisas (CS&I Lab.), Instituto Nacional de Telecomunicações - Inatel
thamer.reis@get.inatel.br, isaque.h@ges.inatel.br, evandro.cesar@inatel.br, felipe.figueiredo@inatel.br

***Abstract*—This article aims to demonstrate the developing and implementing of an autoencoder for digital communication systems in a software-defined radio environment to replace conventional systems. Therefore, concepts of neural networks applied in telecommunications are used to implement a QAM (Quadrature Amplitude Modulation) encoder and a decoder in a GNU Radio Companion environment for a feasibility study in terms of bit error rate (BER).**

***Index Terms*—Autoencoder; software-defined radio; neural networks; telecommunications; BER**

***Resumo*—Esse trabalho visa demonstrar o desenvolvimento e implementação de um *autoencoder* para sistemas de comunicação digital em ambiente de rádio definido por *software* para substituição de sistemas convencionais. Logo, utilizam-se conceitos de redes neurais aplicados em telecomunicações para implementar um *encoder* e um *decoder* QAM em ambiente GNU Radio Companion para estudo de viabilidade em termos de taxa de erro de bit (*bit error rate*, BER).**

***Palavras chave*—Autoencoder; radio definido por software; redes neurais; telecomunicações; BER**

I. Introdução

As comunicações surgiram a partir da necessidade do ser humano de passar informação uns aos outros, possibilitando sua convivência em sociedade e provendo entendimentos e regras entre os envolvidos [1]. Com o homem vivendo em sociedade, surgiu a necessidade de não apenas interagir com quem está perto mas também com quem está longe, assim surgiram as telecomunicações. O termo telecomunicação define o ato de comunicar-se à distância, podendo ser tanto uma comunicação com fio quanto sem fio. Para uma comunicação sem fio, verificam-se vários desafios relacionados ao efeito do canal no sistema de comunicação. O estudo e caracterização dos meios de comunicação sem fio possibilitou o desenvolvimento de sistemas transceptores sofisticados, cujos equipamentos se adaptam às condições do canal de comunicação por meio de técnicas de transmissão e/ou recepção flexíveis para mitigar seu efeito na propagação do sinal. São exemplos de equipamentos e funcionalidades adaptáveis e/ou reconfiguráveis nos transceptores: rádios definidos por *software* (*software-defined radio*, SDR), rádios cognitivos, esquemas de modulação e codificação adaptativas, técnicas de diversidade e arranjos de antenas com capacidade de reconfiguração e direcionamento de feixes [2, 3, 4, 5].

Tecnologias emergentes como SDR e redes definidas por *software* permitem agregar flexibilidade e interoperabilidade às redes de telecomunicações [6]. A implementação de sistemas de comunicação baseados em SDR oferece a possibilidade de operar diferentes configurações de sistemas sobre um mesmo *hardware* [2]. Por outro lado, com o advento da inteligência artificial aplicada às telecomunicações, consideram-se conceitos de aprendizado de máquina (*machine learning*, ML) e profundo (*deep learning*, DL) na redefinição dos atuais sistemas de comunicação sem fio [7, 8]. Dessa forma, pode-se representar o transmissor e receptor por meio de um algoritmo de rede neural, denominado *autoenconder*, visando obter desempenho ótimo fim-a-fim.

Nesse contexto, esse trabalho visa demostrar o desenvolvimento e implementação de um *autoencoder* para sistemas de comunicação digital em ambiente de SDR. Para isso, foi utilizado o programa de código-aberto GNU Radio Companion e a linguagem de programação Python para o desenvolvimento do projeto. Uma rede neural foi desenvolvida e aplicada em um sistema de comunicação digital no ambiente do GNU Radio Companion, sendo o desempenho comparado a sua versão convencional por meio de taxa de erro de bit (*Bit error rate*, BER). Estruturou-se o artigo em quatro seções. Na Seção II, aborda-se conceitos essenciais sobre redes neurais. Discutem-se conceitos e o desenvolvimento do projeto na Seção III, incluindo a implementação do *autoencoder*, a configuração do sistema de comunicação digital no GNU Radio Companion e a comparação de desempenho com um sistema convencional. Os principais comentários e conclusões, assim como trabalhos futuros são abordados na Seção VI.

II. Fundamentos em redes neurais

Redes Neurais são técnicas computacionais com nós interconectados que funcionam como os neurônios do cérebro humano, trafegando informações de forma análoga a como neurônios biológicos enviam sinais uns para os outros [9]. A unidade básica de uma rede neural é denominada de neurônio, sendo associada em estruturas de camadas para a formação de uma rede. Estabelece-se a conexão entre neurônios por meio de estruturas definidas como pesos. Uma rede neural pode ser composta por múltiplas camadas, conforme visto na Figura 1. A camada de entrada é a responsável pela captação da informação que será processada; a(s) camada(s) intermediaria(s) realizam a maior parte do processamento; a camada de saída permite obter os resultados do processamento realizado pelas camadas anteriores. Camadas intermediarias também são denominadas de camadas ocultas. Geralmente, os

valores gerados pelos nós intermediários não são passiveis de uma interpretação lógica.

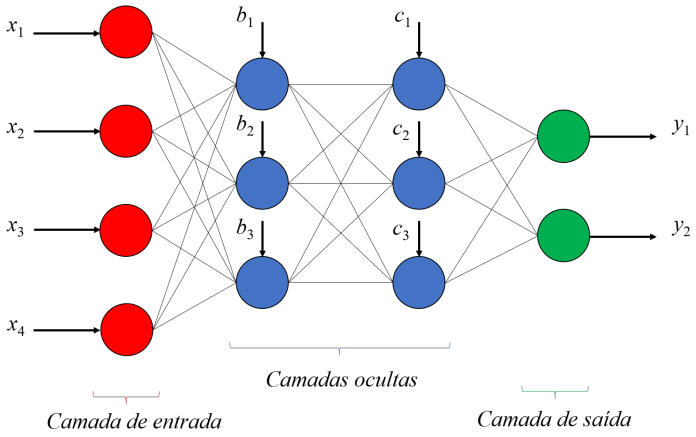

Fig. 1. Exemplo de estrutura de uma rede neural com múltiplas camadas.

O uso de uma rede neural para um determinado propósito requer o seu treinamento com uma base de dados suficiente para se obter resultados com acurácia. O treinamento de uma rede neural pode ser feito através de técnicas de aprendizado supervisionado, não supervisionado e aprendizado por reforço. Para entender o processo de aprendizado, consideram-se os nós de uma rede neural, para fins didáticos, como um valor que varia entre 0 e 1, dependendo da função de ativação. Esse valor é calculado a partir de uma expressão de regressão linear [10], que significa que quanto mais perto do 1 mais influente é o nó. Para cada nó, tem-se pesos diferentes que serão adaptados pelo algoritmo para obter a solução esperada. Considerando x_i o vetor de entrada do nó em análise, w_i o vetor de pesos e $bias$ um número adicionado (positivo ou negativo) para se chegar no valor necessário para o algoritmo. O valor que representa o nó é dado por:

$$\sum_{i=1}^{m} w_i x_i + bias = w_1 x_1 + w_2 x_2 + w_3 x_3 + bias. \quad (1)$$

A expressão em (1) é limitada para problemas complexos, introduzindo a necessidade de operar com uma função de ativação. Uma rede neural sem função de ativação é essencialmente apenas um modelo de regressão linear. A função de ativação faz a transformação não-linear nos dados de entrada, tornando a rede neural capaz de aprender e executar tarefas mais complexas [11]. Os pesos do vetor w_i de cada camada devem ser atualizados ao longo do treinamento para cada iteração. Geralmente, utiliza-se um processo de retropropagação (*backpropagation*) do erro gerado pela rede neural na camada de saída quando comparando o resultado obtido ao fim de uma iteração e o resultado esperado.

No aprendizado supervisionado, deve-se fornecer ao algoritmo dados de treino. Esses dados correspondem a uma amostra que precisa ter possíveis entradas e saídas esperadas. Desta forma o algoritmo começará a alterar e dar valor aos parâmetros em (1), assim conseguindo a partir de uma lógica própria e usando os conceitos de *backpropagation* a interpretar os valores da entrada e chegar aos valores esperados da saída, treinando a(s) camada(s) intermediaria(s). Com o algoritmo treinado, utilizam-se amostras de teste para verificar a acurácia do treinamento.

O aprendizado não supervisionado é um aprendizado na qual os dados de entrada não foram rotulados, classificados ou categorizados previamente [12]. Nesse tipo de aprendizado cria-se regras e separa os dados em grupos de acordo com suas similaridades e diferenças. Algumas técnicas são utilizadas para a realização dessa separação dos grupos, são elas: clusterização, detecção de anomalias, mineração de associação, entre outras [13]. Outra aplicação para os aprendizados não supervisionados é a criação de autoencoders, que será aprofundado na próxima seção. Utilizando uma dessas técnicas o algoritmo é capaz de categorizar os dados, mesmo sem conhecimento prévio da informação.

No aprendizado por reforço, o sistema de inteligência artificial funciona a partir de recompensas. O sistema de aprendizado, chamado de agente, observa o ambiente (estado) e executa ações. Cada ação pode gerar recompensas ou penalidades e o objetivo do agente é maximizar a recompensa recebida por cada ação tomada. Esse tipo de aprendizado é utilizado comumente na criação de algoritmos para jogos e também é amplamente utilizado para ensinar robôs a andar.

No processo de implmentação de um algoritmo de rede neural, devem-se definir suas características por meio de hiperparâmetros, tais como número de camadas, quantidade de nós em cada camada, função de ativação, otimizadores, entre outros [14]. A vantagem de otimizar esses parâmetros é que com os hiperparâmetros otimizados o modelo tende a ter um erro muito menor. Para isso existem diferentes técnicas de otimização, cada uma com suas vantagens e desvantagens. A fim de exemplificar o conceito, primeiramente iremos evidenciar os hiperparâmetros.

O número de nós e camadas são hiperparâmetros importantes, pois relacionam-se ao tempo de processamento da informação. Logo, deve-se ter cuidado pois redes com muitas camadas e neurônios podem levar ao *overfitting*. Esse termo refere-se a quando o algoritmo aprende demais sobre os dados de teste (quase como se decorasse), fazendo com que o algoritmo tenha um desempenho excelente com os dados de treino e precário com dados de testes. Consequentemente, o algoritmo não é capaz de generalizar para outros dados distintos daqueles utilizados no treinamento. A fim de evitar tal efeito é recomendado uma amostra grande de dados de treinamento, um pré-processamento nos dados caso tenha ruído (valores extremos ou incorretos). Outra solução, caso ainda esteja ocorrendo *overfitting*, é a simplificação do algoritmo, com um modelo mais simples.

A função de ativação tem o papel de definir se a informação que o neurônio está recebendo é relevante ou não para o processamento, realizando a ativação ou não do neurônio. Existem diversos tipos de função de ativação como, por exemplo, as funções ReLu (*rectified linear unit*) e Sigmoide. Outro hiperparâmetro importante são os otimizadores, que atualizam os parâmetros de peso em (1) para minimizar a função de perda. A função de perda atua levando a função em direção ao mínimo global, onde o erro será o menor possível. De

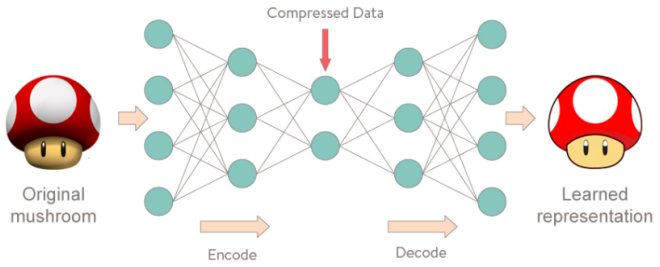

Fig. 2. Exemplo de *autoencoder* [11].

modo geral, isso é feito encontrando o gradiente descendente da função, que indica qual "direção" seguir para que seu valor mínimo seja encontrado rapidamente. Para isso tem-se alguns tipos de otimizadores como SGD, ADAM, ADAGRAD, entre outros.

III. Implementação de um sistema de comunicação digital utilizando *autoencoders*

Autoenconders são redes neurais que empregam aprendizado não supervisionado. Um *autoencoder* tem o objetivo de copiar a entrada para a saída, porém o que realmente é significativo para esse estudo é quando utiliza-se o *autoencoder* de forma incompleta. Essa técnica visa compactar determinado dado da entrada. Utilizando os conceitos de redes neurais, um *autoencoder* incompleto é composto pela compactação do número de neurônios a partir da entrada (processo conhecido como *encode*), passando por um "gargalo"(que é o meio do processo, em que temos o menor número de neurônios na rede) e depois a descompactação dos neurônios resultando na saída [11]. É chamada de incompleta pelo fato das camadas intermediarias terem menos neurônios que as camadas de entrada e saída.

A compressão do dado leva o *autoencoder* precisa em aprender as partes mais importantes do dado de entrada necessárias para a reconstrução, assim outras informações menos significantes não são utilizadas e desconsideradas na saída. Desta forma, tem-se uma representação mais simples da entrada sendo exibida na saída, conforme visto na Figura 2. Utilizando esse modelo aplicado aos conceitos de telecomunicações, pode-se utilizar um *autoencoder* para substituir um transmissor e um receptor convencional.

A. Definições e treinamento do autoencoder

Como o *autoencoder* tem como objetivo substituir um sistema de telecomunicação convencional, o primeiro passo para sua criação é a definição de quais partes do sistema serão substituídas. Para este trabalho, a rede neural irá substituir o mapeador e o demapeador de uma constelação QAM (*Quadrature Amplitude Modulation*). Essa modulação foi escolhida por possuir menos símbolos e facilitar a análise em um primeiro momento. Para a implementação do *autoencoder* , utilizou-se a linguagem de programação em Python e a biblioteca de código aberto TensorFlow (versão 2) para criação e treinamento da rede neural. Definiu-se como hiperparâmetros a função de ativaçao ReLu, otimizador Adam, número de camadas igual a 5 e número de neurônios no gargalo igual a 2 [15]. O gargalo do sistema possui somente dois nós, sendo utilizados como forma de transformar os valores dos nós do gargalo em símbolos complexos para serem transmitidos. O processo de treinamento empregou o Google Colab [16]. Foram utilizadas no total 21000 *epochs*, separadas em três processos o primeiro com 1000 e os outros dois com 10000. O tamanho do *batch* também varia, sendo os dois primeiros 100 e do último 1000. Ao fim do treinamento, a rede neural foi salva para implementação em um sistema de comunicação digital no ambiente do GNU Radio.

B. Implementação do autoencoder no GNU Radio Companion

GNU Radio Companion é um kit de ferramentas de desenvolvimento de *software* gratuito e de código aberto que fornece blocos de processamento de sinal para implementar sistemas de comunicação em SDR [17]. Pode ser utilizado para passar o código para as placas de SDR ou sem o *hardware* a fim de se fazer somente uma simulação. Seu funcionamento é baseado em blocos, que possuem funções distintas e se conectam para criar um sistema. No GNU Radio é possível criarmos blocos funcionais que não sejam padrão do *software*, utilizou-se essa função para as implementações do transmissor e receptor neural, esse blocos podem ser tanto programados em Python quanto em C++.

Para implementação do *autoencoder* dentro do GNU Radio, utilizou-se apenas a parte da rede neural necessária para a função. Foram criados dois blocos, o transmissor contendo a parte do *encoder* e o receptor contendo o *decoder*. Além disso foram necessários outros processamentos dentro dos blocos. No caso do transmissor, colocou-se um número inteiro de 0 a 3 na entrada do *encoder* e na saída converteu-se o valor dos dois últimos neurônios para um número complexo que representa o simbolo referente a entrada. Já no caso do receptor recebeu-se o número complexo e colocaram-se os valores das respectivas posições nos nós do gargalo do sistema na entrada do receptor. Na saída do *decoder* tem-se um vetor de 4 posições (onde cada mensagem é representada por uma posição) com os valores variando de mais infinito a menos infinto, o *decoder* estima a partir da posição que possui o maior número qual a mensagem foi transmitida. O sistema implementado pode ser visto na Figura 3.

C. Resultados e discussões

Para avaliar o desempenho do *autoencoder*, implementou-se o sistema de comunicação digital convencional visto na Figura 4. Empregou-se como métrica de desempenho a BER obtida na saída dos sistemas propostos em função da relação sinal-ruído (signal-to-noise ratio, SNR). Ambos os sistemas operam com um canal AWGN (*Additive white Gaussian noise*). Para o sistema convencional, tem-se uma fonte de bits aleatórios que fornece dados ao mapeador QAM que fornecerá o simbolo, esse simbolo é contaminado com ruído AWGN que será recebido e estimado no receptor. Para o sistema com uso de *autoencoder*, troca-se o mapeador QAM pelo bloco transmissor (*encoder*) e a parte de recepção e feita pelo bloco receptor (*decoder*). O bloco *Constellation Sink* permite observar as constelações de cada sistema, conforme visto na Figura 5. observa-se que a constelação gerada pela

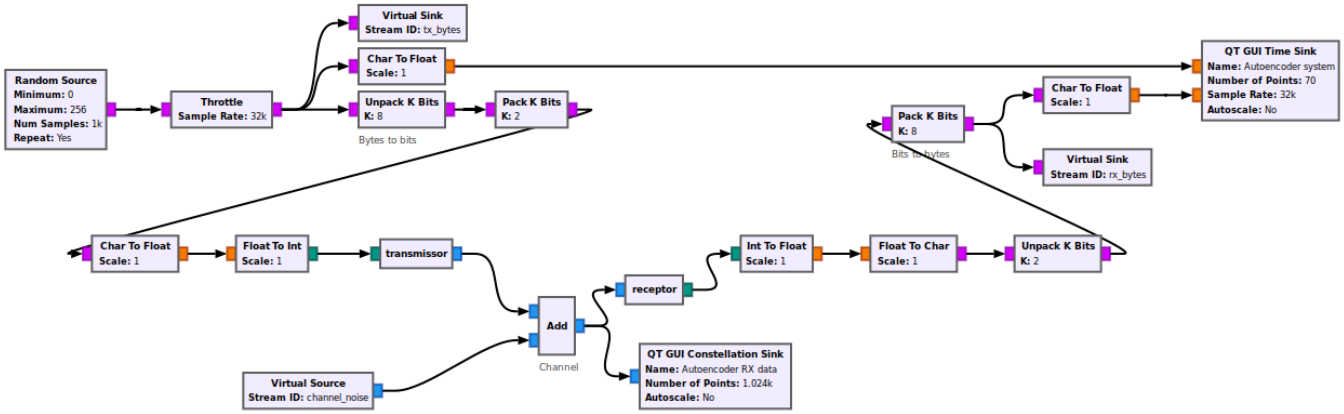

Fig. 3. Sistema de comunicação digital implementado por meio de *autoencoder*.

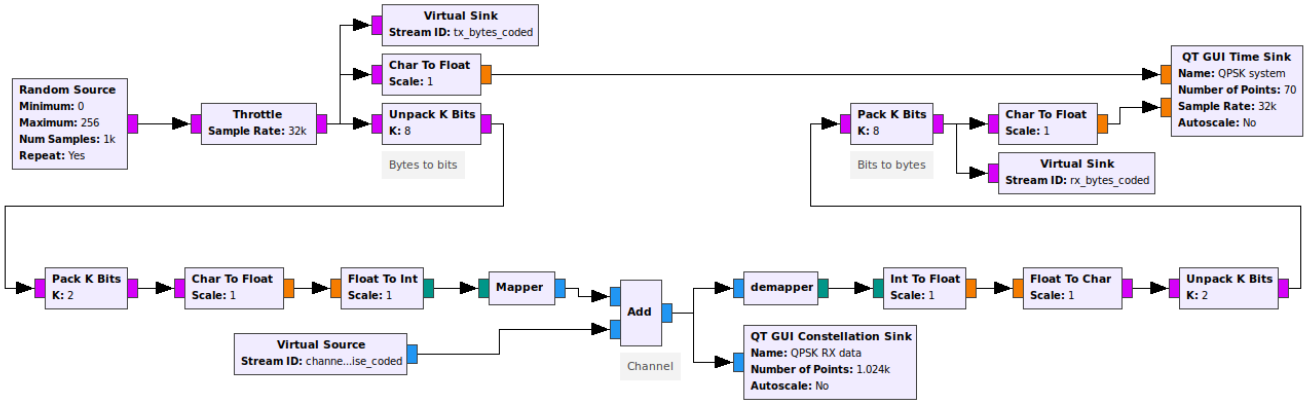

Fig. 4. Sistema de comunicação digital convencional.

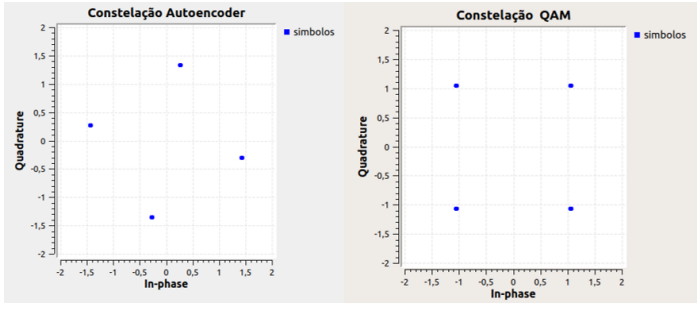

Fig. 5. Comparação entre constelações

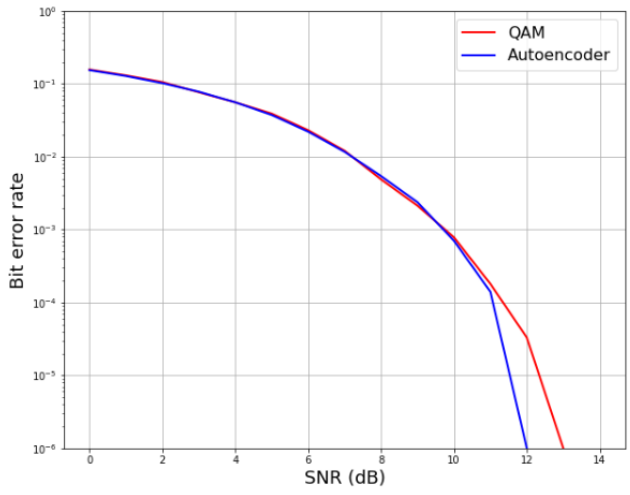

Fig. 6. Curvas de BER para sistema de comunicação digital implementado com uso de *autoenconder* (azul) e convencional (vermelho).

rede neural é diferente do padrão QAM utilizando, sendo oriunda da adaptação da rede neural ao canal AWGN. A partir desses sistemas e de blocos específicos do GNU Radio, foi possível obter os valores e plotar a curva de BER para comparar o desempenho dos dois sistemas. As curvas de BER, vistas na Figura 6 são praticamente idênticas para SNR menor que 11 dB, demonstrando que a constelação proposta pela rede neural possui desempenho similar à constelação QAM convencional.

IV. Conclusão

Neste artigo foi apresentada uma solução utilizando inteligência artificial para substituir um mapeador e demapeador

QAM em um sistema de comunicação digital em ambiente do GNU Radio. Utilizou-se uma rede neural do tipo *autoencoder* para a implementação em ambiente do GNU Radio, sendo o desempenho comparado ao de um sistema convencional por meio de desempenho em BER. Os resultados demonstraram que o *autoencoder* implementado apresenta BER similar ao sistema convencional, não justificando sua implementação. Entretanto, os estudos preliminares conduzidos nesse trabalho visam contribuir para a introdução de conceitos de inteligência artificial em sistemas de telecomunicações com o objetivo de melhorar métricas de desempenho e trazer aspectos de adaptação aos diferentes cenários de operação. Trabalhos futuros visam aprofundar o estudo apresentado para que a rede neural proposta possa superar o sistema convencional em requisitos de BER, justificando sua implementação.

AGRADECIMENTOS

Os autores agradecem ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pela concessão da bolsa, ao Instituto Nacional de Telecomunicações (Inatel) e ao Laboratório de Segurança Cibernética e Internet das Coisas (CS&I Lab.) pelo suporte técnico oferecido.

REFERÊNCIAS

[1] Álvaro Gonçalves de Barros; Carlos Henrique Medeiros de Souza; Risiberg Teixeira. *Evolução das comunicações até a internet das coisas: a passagem para uma nova era da comunicação humana*.

[2] T. F. Collins. *Software-Defined Radio for Enginners*.

[3] V. Ramani; S. K. Sharma. *Cognitive radios: A survey on spectrum sensing, security and spectrum handoff*.

[4] G. Han; L. Sun; G. Yang. *Design of High Gain Cosecant Beam-Forming Array Antenna*.

[5] S. K. Sharma. *Design and development of some novel phased arrays and anti-jamming antennas*.

[6] D. F. Macedo. *Programmable Networks—From Software-Defined Radio to Software Defined Networking*.

[7] A. Anderson; S. R. Young; T. P. Karnowski; J. M. Vann. *Deepmod: An Over-the-Air Trainable Machine Modem for Resilient PHY Layer Communications*.

[8] T. Almeida; E. M. Hung. *Utilização de Técnicas de Aprendizado de Máquina para Demodulação de Sinais*.

[9] ICMC USP. *Redes Neurais Artificiais*. URL: https://sites.icmc.usp.br/andre/research/neural/ (acesso em 30/08/2022).

[10] IBM Cloud Learn Hub. *Redes neurais*. URL: https://www.ibm.com/br-pt/cloud/learn/neural-networks (acesso em 30/08/2022).

[11] Data Science Academy. *Deep Learning Book*. URL: https://www.deeplearningbook.com.br (acesso em 30/08/2022).

[12] TIBCO. *O que é aprendizagem não supervisionada?* URL: https://www.tibco.com/pt-br/reference-center/what-is-unsupervised-learning%5C#:~:text=O%5C%20aprendizado%5C%20n%5C%C3%5C%A3o%5C%20supervisionado%5C%20%5C%C3%5C%A9,tentar%5C%20entender%5C%20por%5C%20conta%5C%20pr%5C%C3%5C%B3pria. (acesso em 30/08/2022).

[13] ALIGER. *Machine learning: entenda o que é aprendizado não supervisionado*. URL: https://www.aliger.com.br/blog/machine-learning-entenda-o-que-e-aprendizado-nao-supervisionado/ (acesso em 03/01/2022).

[14] Arthur Lamblet Vaz. *Otimizando hiperparametros*. URL: https://medium.com/data-hackers/otimizando-os-hiperpar%5C%C3%5C%A2metros-621de5e9be37%5C#:~:text=Hiperpar%5C%C3%5C%A2metros%5C%20s%5C%C3%5C%A3o%5C%20par%5C%C3%5C%A2metros%5C%20de%5C%20modelos,melhor%5C%20acur%5C%C3%5C%A1cia%5C%20em%5C%20seu%5C%20modelo. (acesso em 03/01/2022).

[15] Sebastian Cammerer; Sebastian Dörner; Adriano Pastore. *Learning to Communicate: Hands-on Coding*. URL: https://mlc.committees.comsoc.org/tag/autoencoders/ (acesso em 30/08/2022).

[16] Google. *Colab*. URL: https://research.google.com/colaboratory/intl/pt-BR/faq.html (acesso em 30/08/2022).

[17] GNU Radio. URL: https://www.gnuradio.org/about/ (acesso em 30/08/2022).