Otimizando o Treinamento e a Topologia de um Decodificador de Canal baseado em Redes Neurais

Marcelo Vinícius Cysneiros Aragão, Samuel Baraldi Mafra e Felipe Augusto Pereira de Figueiredo

***Resumo*— Dada sua facilidade de prototipagem e versatilidade, redes neurais vem sendo aplicadas com sucesso à decodificação de canal nas últimas décadas. Após revisar diversas implementações da literatura, este trabalho propõe a análise de aspectos relacionados ao treinamento e à topologia destas redes. Apontando um modelo simples e eficaz para a tarefa, os resultados promissores – comparáveis a um decodificador MAP – ressaltam a aplicabilidade deste tipo de abordagem e aponta possibilidades para estudos futuros.**

***Palavras-Chave*— Decodificação de canal, redes neurais, simulação.**

***Abstract*— Given their ease of prototyping and versatility, neural networks have been successfully applied to channel decoding in recent decades. After reviewing several implementations in the literature, this work proposes the analysis of aspects related to the training and topology of these networks. Pointing to a simple and effective model for the task, the promising results – comparable to a MAP decoder – highlight the applicability of this type of approach and point out possibilities for future studies.**

***Keywords*— Channel decoding, neural networks, simulation.**

I. Introdução

A concepção de decodificadores de canal eficientes é um importante desafio nas telecomunicações. Como quase todas as transmissões digitais são protegidas por códigos corretores de erro, mesmo pequenas melhorias no ganho de codificação, taxa de transferência ou custo de implantação podem ter grandes impactos no processo fim-a-fim [1]. Uma das abordagens para este problema é a implementação de decodificadores baseados em redes neurais (ou "decodificadores neurais").

No final da década de 80, Bruck e Blaum [2] enunciaram que a "*decodificação de um código de bloco linear corretor de erros é equivalente a encontrar um máximo global da função de energia de uma determinada rede neural*". No mesmo ano, Zeng, Hush e Ahmed [3] implementaram um decodificador de máxima verossimilhança usando uma rede neural, que pode ser treinada em tempo polinomial (o que é vantajoso, se comparado ao cômputo de todos os códigos de Hamming, o que demanda complexidade temporal exponencial) e cuja arquitetura pode ser generalizada para decodificar qualquer código de bloco (não necessariamente linear).

Marcelo Vinícius Cysneiros Aragão, Departamento de Engenharia de Computação e Software, Instituto Nacional de Telecomunicações, Santa Rita do Sapucaí-MG, e-mail: marcelovca90@inatel.br; Samuel Baraldi Mafra, Departamento de Engenharia de Telecomunicações, Instituto Nacional de Telecomunicações, Santa Rita do Sapucaí-MG, e-mail: samuelbmafra@inatel.br; Felipe Augusto Pereira de Figueiredo, Departamento de Engenharia de Telecomunicações, Instituto Nacional de Telecomunicações, Santa Rita do Sapucaí-MG, e-mail: felipe.figueiredo@inatel.br.

Nas duas décadas seguintes, Caid e Means [1] implementaram um decodificador corretor de código usando uma rede neural, identificando vantagens desta abordagem em cenários nos quais as suposições de ruído AWGN (*Additive White Gaussian Noise*) e canal BSC (*Binary Symmetric Channel*) são violadas. Tallini e Cull [4] e Wu, Tseng e Huang [5] também propuseram implementações, ressaltando a aplicabilidade desta abordagem. Di Stefano et al. [6] examinaram as possíveis soluções para o problema de codificação de Hamming usando redes do tipo *Counter Propagation* e *Back Propagation*, e Abdelbaki, Gelenbe e El-Khamy [7] usaram redes neurais aleatórias. Neste caso, a implementação em *hardware* foi um ponto positivo, sugerindo a criação de *chips* genéricos capazes de resolver uma vasta gama de problemas, bastando ser carregados com redes já treinadas.

Mais recentemente, Nachmani, Be'Ery e Burshtein [8] sugeriram o emprego de redes neurais profundas, que incluem como vantagens a capacidade de ter seu desempenho melhorado mesmo após o treinamento e o aprendizado simultâneo tanto do canal quanto do código linear. Também destaca-se o trabalho de Lyu et al. [9], que compara decodificadores baseados em redes *feedforward*, convolucionais e recorrentes. No caso, o modelo de rede profunda do primeiro apresentou desempenho análogo à recorrente do segundo.

Um trabalho particularmente relevante a este é o de Gruber et al. [10], que avaliaram o emprego de redes neurais profundas na decodificação de códigos aleatórios e estruturados, visando verificar se os estruturados são mais fáceis de serem "aprendidos" do que os aleatórios, e se tais redes são capazes de decodificar palavras-código nunca vistas em seu treinamento.

Por fim, Xu et al. [11] propuseram um modelo híbrido baseado em duas redes neurais: uma convolucional, responsável pela equalização do sinal recebido (isto é, minimização da deterioração de canal e distorção não linear), e uma profunda, responsável pela decodificação do sinal.

Comprovada a relevância do tema, o objetivo deste trabalho consiste, portanto, em estender a experimentação de Gruber et al. [10], avaliando diferentes otimizadores, hiperparâmetros e topologias de redes neurais, de modo a conceber um decodificador neural ainda mais preciso e robusto.

O trabalho está organizado da seguinte forma: na seção I é feita uma introdução sobre o problema e são visitadas diversas abordagens propostas na literatura para resolvê-lo. Em seguida, na seção II, é introduzido e explicado o modelo do sistema, utilizado na condução dos experimentos discriminados na III. Os resultados experimentais são apresentados e discutidos na seção IV. Por fim, a seção V conclui o presente trabalho, recapitulando seus principais pontos.

II. Modelo do Sistema

Para simular computacionalmente o processo de codificação, transmissão, recepção e decodificação, foi utilizado o *setup* experimental ilustrado na Figura 1, cujos blocos serão explicados na sequência.

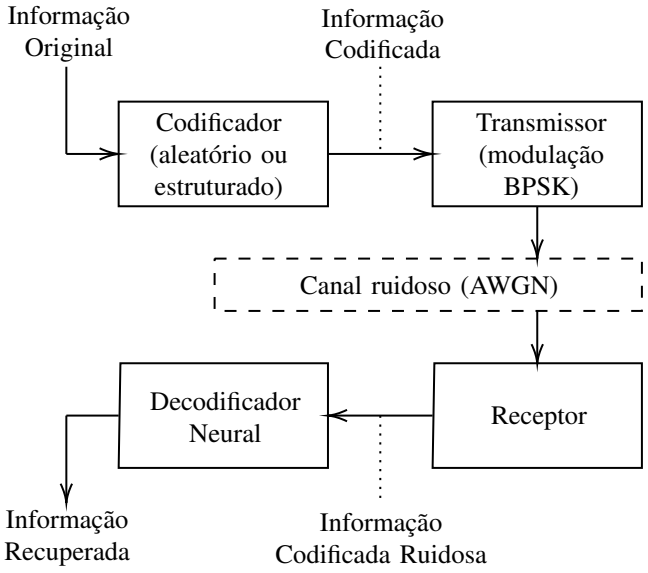


Fig. 1: Diagrama de blocos do *setup* experimental.

No transmissor, a informação é codificada em uma palavra-código de comprimento e taxa de código fixos. Os *bits* codificados são submetidos a um modulador BPSK (*Binary Phase Shift Keying*) e transmitidos através de um canal AWGN.

No receptor, a mensagem ruidosa recebida é decodificada, visando a recuperação da informação original. Para tal, foi empregada uma rede neural do tipo *feedforward* com $N = 16$ entradas (uma para cada bit dos códigos polares ou aleatórios com distância de Hamming superior a 2 [10]), quantidades variáveis de camadas escondidas e de neurônios em cada camada e função de ativação ReLU (*Rectified Linear Unit*). A Figura 2 ilustra uma rede deste tipo, com uma camada de entrada (com duas entradas), duas camadas ocultas (com quatro e dois neurônios, respectivamente) e uma camada de saída (com uma única saída).

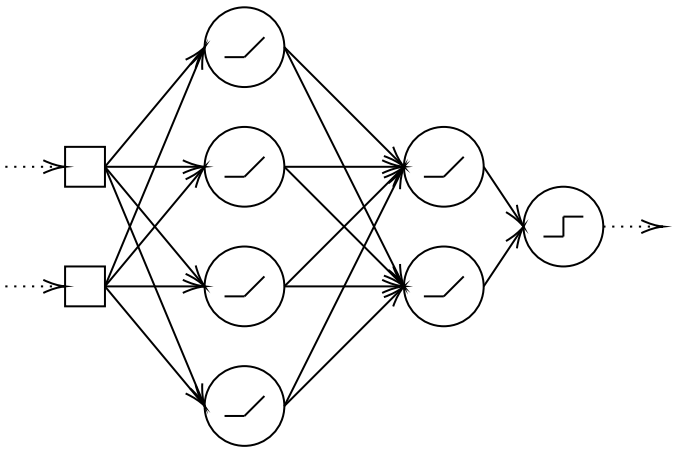


Fig. 2: Exemplo de rede neural com duas camadas ocultas.

III. Experimentos

Visando verificar a viabilidade de decodificadores neurais, bem como avaliar a influência de hiperparâmetros e topologias no desempenho do decodificador, os experimentos deste trabalho foram separados em duas categorias, explicadas a seguir.

A. Avaliação de diferentes quantidades de épocas de treinamento

Esta categoria de experimentos visa avaliar o desempenho (em termos de BER para diversas taxas de E_b/N_0 (relação sinal-ruído normalizada) e tomando como base o decodificador MAP – *Maximum a Posteriori*) do decodificador neural quando submetido a diferentes quantidades de épocas de treinamento. A tabela I sumariza os experimentos desta categoria.

TABELA I: Sumarização da primeira família de experimentos.

Exp.	Código	Topologia da rede neural	Parâmetros	Épocas (2^n)
1	Polar	128-64-32 [10]	12.776	18
2				20
3				22
4	Aleatório			18
5				20
6				22

B. Avaliação de diferentes topologias de rede neural

Esta categoria de experimentos, por sua vez, tem o objetivo de avaliar o desempenho (novamente, em termos de BER para diversas taxas de E_b/N_0 e tomando como base o decodificador MAP) do decodificador neural quando treinado com diferentes otimizadores, sendo eles submetidos a diferentes topologias de redes neurais (i.e., mais quantidades de camadas escondidas, com mais neurônios em cada uma delas) . A tabela II sumariza os experimentos desta categoria.

TABELA II: Sumarização da segunda família de experimentos.

Exp.	Código	Topologia da rede neural	Quantidade de parâmetros	Épocas (2^n)
7	Polar	256-128-64-32	47.848	20
8		512-256-128-64-32	183.528	
9		1024-512-256-128-64-32	717.032	
10	Aleatório	256-128-64-32	47.848	
11		512-256-128-64-32	183.528	
12		1024-512-256-128-64-32	717.032	

O treinamento da rede neural foi feito considerando diversos otimizadores (mais especificamente, Adadelta [12], Adagrad [13], Adam [14], Adamax [14], Nadam [15], RMSprop [16] e SGD [17, 18]), cuja função é minimizar uma função de erro (neste caso, o MSE – *Mean Squared Error*) por meio do ajuste dos pesos sinápticos da rede.

Em cada experimento, o decodificador neural foi treinado com todos os otimizadores e, em seguida, foi calculada a BER para 11 valores de E_b/N_0 (variando de 0 até 10 dB). Foram consideradas 2×10^6 palavras-código de 16 $bits$ de comprimento e taxa de código de 50%. Os resultados experimentais e as discussões pertinentes são apresentados na sequência.

IV. Resultados e Discussão

Os experimentos, discriminados nas tabelas I e II, foram conduzidos em um computador com processador AMD Ryzen™ 9 5950X, 128 GB de RAM DDR4-3200, placa de vídeo Nvidia GeForce® RTX 3090 24GB GDDR6X (*driver* versão 512.59) e sistema operacional Microsoft Windows 11 Education x64 (versão 21H2).

Considerando as categorias de experimentos elucidadas nas seções III-A e III-B, o resultado esperado é que seja possível conceber um decodificador baseado em rede neural que seja capaz de produzir uma BER por E_b/N_0 satisfatória (i.e. próxima ao decodificador MAP) sem a necessidade de uma quantidade de épocas elevada e/ou topologia demasiadamente complexa, já que tais fatores demandariam tempos de treinamento e poder computacional impraticáveis.

Para facilitar a análise dos resultados e verificar se os resultados esperados foram alcançados, ela foi dividida em subseções que focam em aspectos distintos (porém complementares) da experimentação, e que serão apresentadas a partir deste ponto do trabalho.

A. Influência da quantidade de épocas

Os resultados das execuções dos experimentos 1–6 são ilustrados nas Figuras 3a–3f, respectivamente. As seguintes conclusões podem ser obtidas a partir deles:

- No caso do código polar (Figuras 3a, 3b e 3c) nota-se que, adotando uma topologia fixa (i.e. com 3 camadas ocultas), o aumento no número de épocas de treinamento (de 2^{18} para 2^{20}) proporcionou uma melhoria considerável no desempenho dos otimizadores. Isso fica evidente no caso do Adadelta e SGD, pois suas BERs deixaram de apresentar uma tendência de estabilização e passaram a decrescer. Outro ponto importante é que, mesmo com apenas 2^{18} épocas, os otimizadores Nadam, Adam e Adamax já apresentavam desempenho próximo ao MAP, que continuou a melhorar com o aumento de épocas.
- No caso do código aleatório (Figuras 3d, 3e e 3f), apesar de o aumento do número de épocas melhorar o desempenho de todos os otimizadores (o que pode ser percebido com a aproximação de suas curvas de BER em relação à do decodificador MAP), o decodificador neural ainda não apresenta desempenho satisfatório.

Até o momento, constata-se que, mesmo um decodificador neural de topologia "simples" e treinado com uma quantidade "média" (i.e. 2^{20}) de épocas, já é capaz de apresentar desempenho análogo MAP na decodificação de código polar. Entretanto, para o código aleatório, o resultado não foi satisfatório, o que demanda maior investigação e experimentação.

B. Influência da topologia da rede neural

Os resultados das execuções dos experimentos 7–12 são ilustrados nas Figuras 3g–3l, respectivamente. As seguintes conclusões podem ser obtidas a partir deles:

- No caso do código polar (Figuras 3g, 3j e 3h) nota-se que, adotando uma quantidade fixa de épocas de treinamento (i.e., 2^{20}), a complexificação da topologia da rede neural (de 4 até 6 camadas ocultas) proporcionou uma melhoria generalizada do desempenho dos otimizadores, sendo que a maioria deles apresenta desempenho próximo ao MAP já com 4 camadas (com exceção do Adadelta e SGD).
- No caso do código aleatório (Figuras 3k, 3i e 3l), só foi possível atingir desempenho próximo ao MAP com uma topologia ainda mais complexa avaliada (i.e., com 5 camadas ocultas) e com o otimizador Adamax. Em contrapartida, os otimizadores Adadelta e SGD foram os que apresentaram os piores resultados, novamente.

Neste ponto, constata-se que um código de maior complexidade (neste caso, aleatório) demanda uma rede mais complexa.

C. Comportamento dos tempos de treinamento e teste

Uma outra análise importante que deve ser feita baseia-se na observação dos tempos de treinamento (i.e., o tempo demandado para ajustar os parâmetros da rede) e de teste (i.e., o tempo para calcular a BER em cada um dos pontos de E_b/N_0) em cada cenário. Neste sentido, analisando os valores dispostos na tabela III, percebe-se que:

- o tempo de treinamento cresce linearmente em função do número de épocas, pois conforme o número de épocas quadruplica (de 2^{18} para 2^{20} e de 2^{20} para 2^{22}), o mesmo acontece com o tempo de treinamento;
- o tempo de teste permanece constante, já que a quantidade de informação submetida para calcular a BER em cada ponto de Eb/N_0 é igual (i.e., $16Mbits$);
- a complexificação da topologia da rede neural teve pouco impacto sobre os tempos de treinamento e de teste, mesmo com o considerável aumento da quantidade de parâmetros conforme camadas eram acrescentadas.

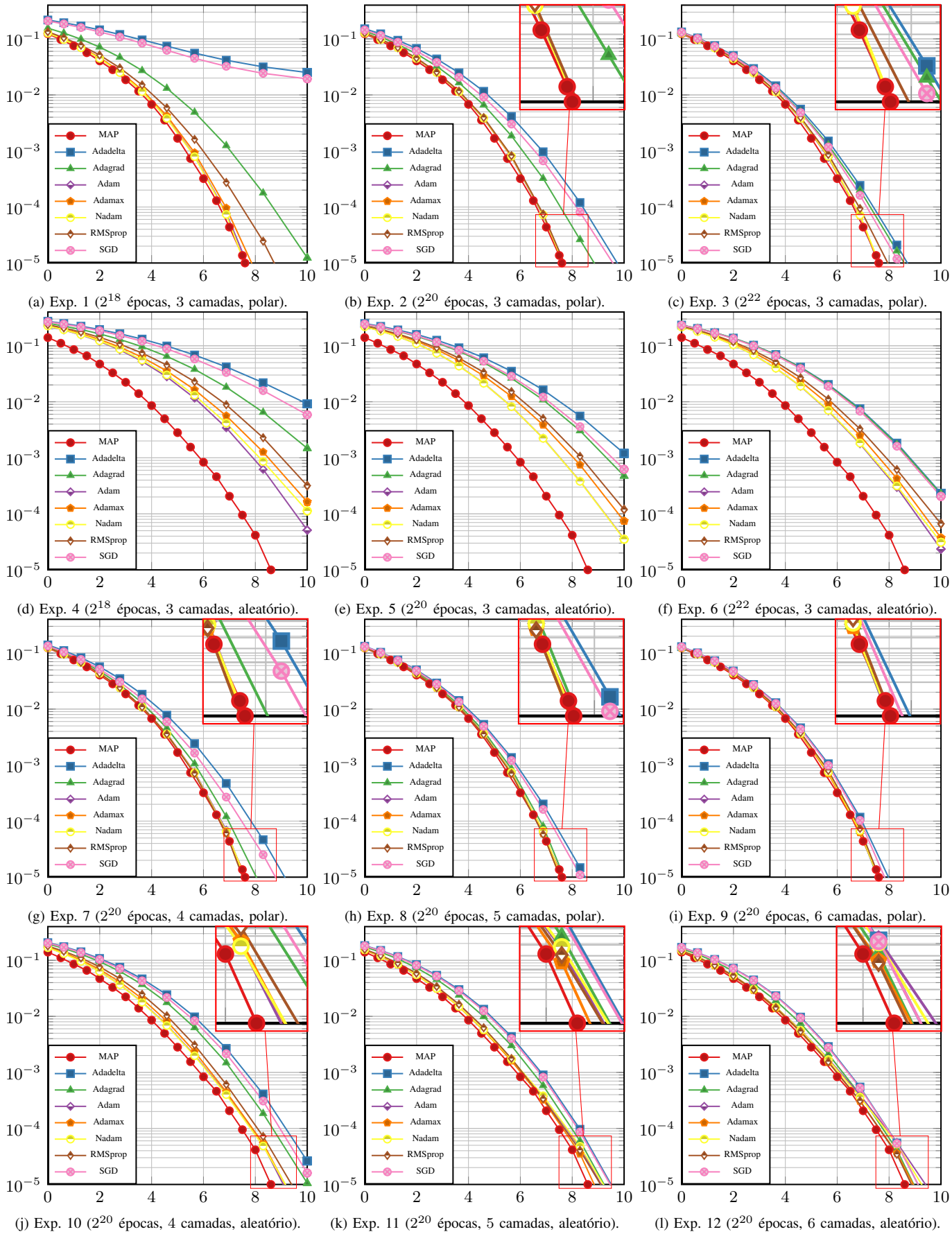
Diante dos resultados obtidos, constata-se que foi possível aplicar com sucesso redes neurais para resolver o problema com desempenho satisfatório (mais especificamente, com 4 camadas e qualquer otimizador – com exceção do Adadelta e SGD – para o código polar, e 5 camadas e otimizador Adamax para o código aleatório, sendo ambas treinadas por 2^{20} épocas). Também observa-se uma maior dificuldade em "aprender" códigos aleatórios do que estruturados, fato também foi apontado por Gruber et al. [10], o que demanda redes mais complexas e, consequentemente, maior custo computacional.

TABELA III: Tempos de treinamento e de testes.

Exp.	Tempo de treinamento (média ± IC @ HH:MM:SS)	Tempo de teste (média ± IC @ HH:MM:SS)
1	00:16:53 ± 00:02:22	00:17:14 ± 00:00:14
2	01:06:31 ± 00:07:33	00:17:44 ± 00:00:11
3	04:13:02 ± 00:23:22	00:16:42 ± 00:00:36
4	00:15:33 ± 00:01:32	00:13:24 ± 00:00:17
5	01:00:11 ± 00:06:31	00:13:06 ± 00:00:13
6	04:01:02 ± 00:25:48	00:13:06 ± 00:00:06
7	01:06:21 ± 00:16:26	00:16:28 ± 00:00:05
8	01:09:24 ± 00:18:17	00:16:17 ± 00:00:09
9	01:11:32 ± 00:17:45	00:16:47 ± 00:00:04
10	01:02:30 ± 00:07:15	00:13:05 ± 00:00:19
11	01:05:46 ± 00:11:04	00:13:03 ± 00:00:20
12	01:06:19 ± 00:08:41	00:13:11 ± 00:00:08

O código e as instruções para reprodução dos experimentos, bem como todos os resultados obtidos, estão disponíveis em um repositório[1] público do GitHub.

[1]https://github.com/marcelovca90/DL-ChannelDecoding



(a) Exp. 1 (2^{18} épocas, 3 camadas, polar). (b) Exp. 2 (2^{20} épocas, 3 camadas, polar). (c) Exp. 3 (2^{22} épocas, 3 camadas, polar).

(d) Exp. 4 (2^{18} épocas, 3 camadas, aleatório). (e) Exp. 5 (2^{20} épocas, 3 camadas, aleatório). (f) Exp. 6 (2^{22} épocas, 3 camadas, aleatório).

(g) Exp. 7 (2^{20} épocas, 4 camadas, polar). (h) Exp. 8 (2^{20} épocas, 5 camadas, polar). (i) Exp. 9 (2^{20} épocas, 6 camadas, polar).

(j) Exp. 10 (2^{20} épocas, 4 camadas, aleatório). (k) Exp. 11 (2^{20} épocas, 5 camadas, aleatório). (l) Exp. 12 (2^{20} épocas, 6 camadas, aleatório).

Fig. 3: Resultados dos experimentos. Em todos os gráficos, o eixo horizontal representa a relação sinal-ruído normalizada (E_b/N_0), e o eixo vertical, a taxa de erro de *bit* (BER) em escala logarítmica.

V. Conclusão

Ao longo das últimas décadas, diversas implementações de decodificadores neurais vêm sendo propostas devido a sua facilidade de prototipagem e alta capacidade de generalização. Os notáveis desempenhos de operação, tanto em termos de precisão quanto de versatilidade, reafirmam o promissor futuro deste tipo de abordagem à tarefa de decodificação de canal.

O presente trabalho teve como objetivo analisar a influência da escolha de otimizadores, da quantidades de épocas de treinamento e da complexidade da topologia da rede neural empregada, no sentido de minimizar a taxa de erro desse tipo de decodificador, sem demandar tempo e/ou recursos computacionais impraticáveis.

Diante da análise dos resultados experimentais, constatou-se que o modelo proposto foi capaz de produzir resultados satisfatórios, tanto em cenários de código estruturado quanto aleatório. Isto ressalta, mais uma vez, a aplicabilidade de modelos de aprendizado de máquina (mais especificamente, redes neurais) à resolução de problemas presentes no dia-a-dia, como a decodificação de canal.

Como sugestões para trabalhos futuros, citam-se o emprego de *autoencoders*, outras melhorias no processo de treinamento (como regularização, *early stopping* e *dropout*) e, por fim, verificar o desempenho do decodificador neural em outras configurações de canal, tipos, comprimentos e taxas de código.

Agradecimentos

Os autores agradecem o apoio financeiro do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), (403827/2021-3), Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) (2021/06946-0) e parcialmente financiado pela Rede Nacional de Ensino e Pesquisa (RNP) com recursos do Ministério da Ciência, Tecnologia e Inovações (MCTIC), concessão número 01245.010604/2020-14, sob o projeto Brasil 6G do Centro de Referência em Radiocomunicações (CRR) do Instituto Nacional de Telecomunicações (Inatel), Brasil.

Referências

[1] William R. Caid e Robert W. Means. "Neural network error correcting decoders for block and convolutional codes". Em: *IEEE Global Telecommunications Conference and Exhibition*. Vol. 2. 1990, pp. 1028–1031. DOI: 10.1109/glocom.1990.116658.

[2] Jehoshua Bruck e Mario Blaum. "Neural Networks, Error-Correcting Codes, and Polynomials over the Binary n-Cube". Em: *IEEE Transactions on Information Theory* 35.5 (1989), pp. 976–987. DOI: 10.1109/18.42215.

[3] Gengsheng Zeng, Don Hush e Nasir Ahmed. "Application of neural net in decoding error-correcting codes". Em: *IEEE International Symposium on Circuits and Systems*. Vol. 2. 1989, pp. 782–785. DOI: 10.1109/iscas.1989.100467.

[4] L G Tallini e P Cull. "Neural nets for decoding error-correcting codes". Em: *IEEE Technical Applications Conference and Workshops*. 1995, p. 89.

[5] Ja-Ling Wu, Yuen-Hsien Tseng e Yuh-Ming Huang. "Neural Network Decoders for Linear Block Codes". Em: *International Journal of Computational Engineering Science* 03.03 (2002), pp. 235–255. DOI: 10.1142/s1465876302000629.

[6] A Di Stefano, O Mirabella, G Di Cataldo e G Palumbo. "On the use of neural networks for Hamming coding". Em: *IEEE International Sympoisum on Circuits and Systems*. 1991, pp. 1601–1604.

[7] Hossam Abdelbaki, Erol Gelenbe e Said E. El-Khamy. "Random Neural Network decoder for error correcting codes". Em: *International Joint Conference on Neural Networks*. Vol. 5. 1999, pp. 3241–3245. DOI: 10.1109/ijcnn.1999.836175.

[8] Eliya Nachmani, Yair Be'Ery e David Burshtein. "Learning to decode linear codes using deep learning". Em: *Allerton Conference on Communication, Control, and Computing*. 2017, pp. 341–346. DOI: 10.1109/ALLERTON.2016.7852251.

[9] Wei Lyu, Zhaoyang Zhang, Chunxu Jiao, Kangjian Qin e Huazi Zhang. "Performance evaluation of channel decoding with deep neural networks". Em: *IEEE International Conference on Communications*. Vol. 2018-May. 2018, pp. 1–6. DOI: 10.1109/ICC.2018.8422289.

[10] Tobias Gruber, Sebastian Cammerer, Jakob Hoydis e Stephan Ten Brink. *On deep learning-based channel decoding*. 2017. DOI: 10.1109/CISS.2017.7926071.

[11] Weihong Xu, Zhiwei Zhong, Yair Beery, Xiaohu You e Chuan Zhang. "Joint neural network equalizer and decoder". Em: *International Symposium on Wireless Communication Systems*. Vol. 2018-Augus. 2018, pp. 1–5. DOI: 10.1109/ISWCS.2018.8491056.

[12] Matthew D. Zeiler. "ADADELTA: An Adaptive Learning Rate Method". Em: *arXiv preprint arXiv:1212.5701* (2012).

[13] John Duchi, Elad Hazan e Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization". Em: *Conference on Learning Theory (COLT)* 12.7 (2010), pp. 257–269.

[14] Diederik P. Kingma e Jimmy Lei Ba. *Adam: A method for stochastic optimization*. 2015.

[15] Timothy Dozat. "Incorporating Nesterov Momentum into Adam". Em: *International Conference on Learning Representations (ICLR)* 1 (2016), pp. 2013–2016.

[16] Kevin Swersky Geoffrey Hinton, Nitish Srivastava. "Neural Networks for Machine Learning Lecture". Em: *COURSERA: Neural networks for machine learning* 4.2 (2012), pp. 26–31.

[17] Herbert Robbins e Sutton Monro. "A Stochastic Approximation Method". Em: *Annals of Mathematical Statistics* 22.3 (1951), pp. 400–407. DOI: 10.1214/aoms/1177729586.

[18] J. Kiefer e J. Wolfowitz. "Stochastic Estimation of the Maximum of a Regression Function". Em: *The Annals of Mathematical Statistics* 23.3 (1952), pp. 462–466. DOI: 10.1214/aoms/1177729392.