Hyperparameter Optimization of a Deep Learning-Based Channel Decoder

Marcelo Vinícius Cysneiros Aragão
marcelo.vinicius@dtel.inatel.br

Nicole Souza
nicole.souza@mtel.inatel.br

Resumo

O uso de decodificadores de canal baseados em redes neurais vem se destacando nas últimas décadas. Vantagens apontadas por esta abordagem incluem o reduzido tempo de treinamento, alta capacidade de generalização e facilidade de implementação em *hardware*. Neste sentido, o presente trabalho descreve a concepção e elucida os principais resultados obtidos por um decodificador neural baseado em redes profundas. Em seguida, levanta possibilidades de pesquisas futuras, dentre as quais, a otimização de hiperparâmetros, que é explorada com experimentos adicionais. Tais simulações concluem que o modelo original pode ser aprimorado com o emprego de um otimizador que inclui um termo de *momentum* mais moderno.

1 Introdução

A concepção de decodificadores 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 (chamados simplesmente de decodificadores neurais). Ao longo das últimas décadas, diversos trabalhos – alguns dos quais serão brevemente revisados a seguir – propuseram arquiteturas que abrangem desde redes *feedforward*, recorrentes e convolucionais, até aquelas baseadas em aprendizado profundo.

É justamente neste contexto que o presente trabalho se encontra. Nele, Gruber et al. [2] avaliaram o emprego de redes neurais profundas na decodificação de códigos aleatórios e estruturados, com o objetivo de verificar se códigos estruturados são mais fáceis de serem "aprendidos" do que códigos aleatórios, e se uma rede neural é capaz de decodificar palavras-código que nunca viu durante seu treinamento.

2 Revisão bibliográfica

Uma das formulações propostas para o problema de decodificação neural data do final da década de 80, quando Bruck e Blaum [3] 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*".

Concretizando esta proposta, Zeng, Hush e Ahmed [4] 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, que apresenta complexidade temporal exponencial) e cuja arquitetura pode ser generalizada para decodificar qualquer código de bloco (não necessariamente linear).

Na década seguinte, Caid e Means [1] implementaram um decodificador corretor de código usando uma rede neural, e indicaram vantagens deste tipo de abordagem em cenários nos quais as suposições de ruído gaussiano branco aditivo (AWGN) e canal simétrico binário (BSC) são violadas. Tallini e Cull [5] e Wu, Tseng e Huang [6] também implementaram decodificadores neurais, ressaltando a aplicabilidade desta abordagem.

Outros tipos de rede também foram propostos, como no trabalho de Di Stefano et al. [7], que examinaram as possíveis soluções para o problema de codificação de Hamming usando redes do tipo Counter Propagation e Back Propagation, e no de Abdelbaki, Gelenbe e El-Khamy [8], que usaram redes neurais aleatórias. Neste caso, a implementação em *hardware* descomplicada foi apontada como uma vantagem no sentido de facilitar a criação de *chips* dedicados a esta tarefa.

Mais recentemente, Nachmani, Be'ery e Burshtein [9] trilharam o caminho para desenvolvimento de decodificadores usando redes neurais profundas. Vantagens desta abordagem incluem a capacidade de melhorar o desempenho da rede mesmo após já ter sido treinada e o aprendizado simultâneo tanto do canal quanto do código linear.

3 Métodos

O *setup* experimental utilizado por Gruber et al. [2]é ilustrado no diagrama da Figura 1, cujos blocos serão explicados na sequência.

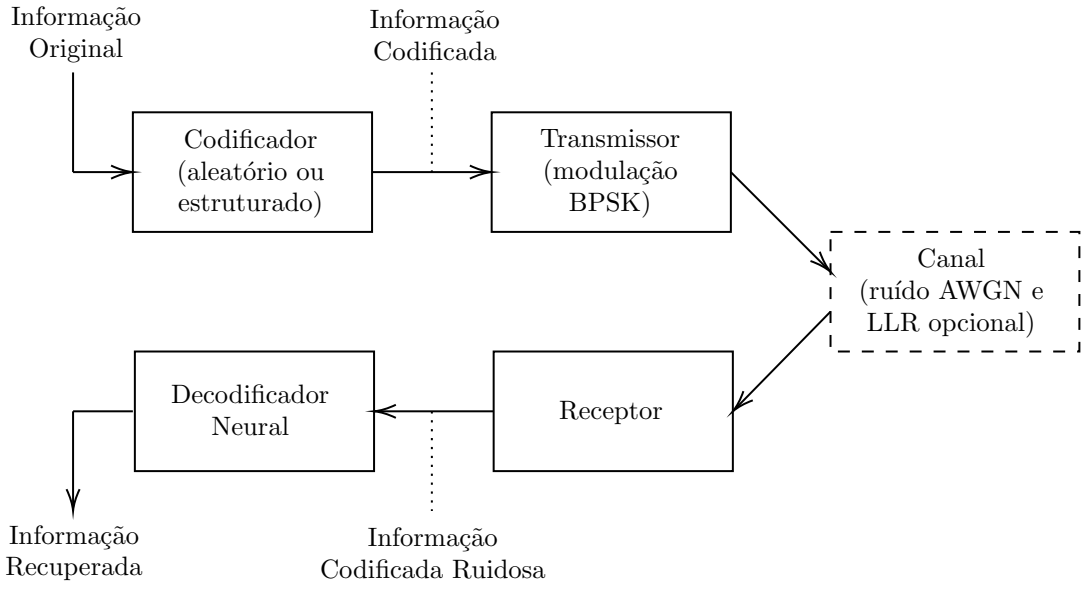


Figura 1: Diagrama de blocos do *setup* experimental.

No lado do transmissor, uma informação é codificada em uma palavra-código de comprimento e taxa de código fixos. Os bits codificados são modulados por um modulador BPSK (*Binary Phase Shift Keying*) e transmitidos por um canal ruidoso AWGN (*Additive White Gaussian Noise*). Opcionalmente, é possível enviar dados de verossimilhança logarítmica (LLR – *Log-Likelihood Ratio*) em vez da saída convencional do canal.

No lado do receptor, a mensagem ruidosa recebida deve ser decodificada para que a informação original seja recuperada. Para tal, foi empregada uma rede neural *feedforward* com três camadas escondidas, função de ativação ReLU (*Rectified Linear Unit*), diferentes quantidades de neurônios por camada e minimização de erro via gradiente descendente estocástico.

Os autores introduzem a métrica NVE (*Normalized Validation Error*) para avaliar desempenho da rede. É dada pela média das razões entre as BER (*Bit Error Rate*) obtidas pelo modelo neural e pela decodificação MAP (*Maximum a Posteriori*) convencional.

Sempre levando em consideração tanto a codificação de código aleatório quanto de estruturado (polar), os seguintes experimentos são conduzidos no trabalho:

1. Influência da SNR (*Signal-to-Noise Ratio*) ($E_b/N_0 = -4, -3, \ldots, +6$ dB) no NVE, durante a etapa de treinamento.
2. Influência da quantidade de épocas de treinamento ($M_{ep} = 2^{10}, 2^{12}, \ldots, 2^{18}$) na BER, considerando:
   (a) Uso (ou não) dos valores de verossimilhança logarítmica na saída do transmissor;
   (b) Emprego de diferentes funções de erro na otimização (MSE – *Mean Squared Error* ou BCE – *Binary Cross-Entropy*);
   (c) Quantidades crescentes de neurônios nas três camadas escondidas do decodificador (128-64-32, 256-128-64, 512-256-128 e 1024-512-256);
3. Escalabilidade do decodificador neural considerando informações de tamanhos crescentes ($k = 8, 9, \ldots, 14$ *bits*) e taxas de código variáveis ($p = 70\%, 80\%, 90\%, 100\%$).
4. Capacidade de decodificar palavras-código inéditas avaliando a BLER (*Block Error Rate*) para diferentes SNRs e particionamentos de dados entre conjunto de treinamento e de testes (100%/0%, 90%/10%, 80%/20% e 70%/30%).

4 Resultados

Após a compilação dos resultados experimentais, Gruber et al. [2] apontaram que:

- O código polar demanda uma SNR de treinamento consideravelmente menor (1 dB) do que o código aleatório (4 dB) para atingir um NVE aceitável;
- Códigos estruturados são de fato mais fáceis de aprender do que códigos aleatórios, ou seja, menos períodos de treinamento são necessários;
- Quando a rede neural é treinada com um grande número de épocas, o emprego de LLR ou valores de canal, a normalização dos valores de entrada e a escolha da função de perda não influenciam no resultado;
- Quanto maior a quantidade de neurônios nas camadas ocultas da rede, menos épocas são necessárias para atingir um NVE aceitável;
- O NVE aumenta exponencialmente conforme o número de *bits* de informação aumenta.
- Quanto menor for conjunto de treinamento, menor será a capacidade de generalização da rede (para quantidades de épocas fixas), especialmente em casos de códigos aleatórios;
- Redes neurais são capazes de generalizar ou "interpolar" para o livro de código completo depois de ter visto apenas um subconjunto de exemplos, sempre que o código tiver estrutura.

5 Discussão

Os resultados experimentais reafirmam a aplicabilidade de decodificadores neurais (mais especificamente, baseado em redes profundas) à decodificação de canal.

Os resultados obtidos por Gruber et al. [2] foram compatíveis com trabalhos posteriores, como o de Lyu et al. [10], que implementaram decodificadores neurais baseados em redes *feedforward* (MLP – *Multilayer Perceptron*), convolucionais (CNN – *Convolutional Neural Network*) e recorrentes (RNN *Recurrent Neural Networks*). No caso, o modelo de rede profunda do primeiro apresentou comportamento da curva BER/E_B/N_0 compatível com a RNN do segundo.

Também observou-se compatibilidade com os resultados de Xu et al. [11], que 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.

6 Conclusões e perspectivas para novos trabalhos

Além de demonstrar a viabilidade do desenvolvimento de decodificadores baseados em redes neurais, o trabalho abriu caminho para pesquisas futuras, pois diversos aspectos podem ser avaliados em trabalhos futuros, sempre visando a obtenção de uma decodificação ainda mais precisa (ou seja, modelos cuja curva BER/E_B/N_0 decai mais rapidamente para SNRs cada vez menores, aproximando-se ao máximo do desempenho MAP).

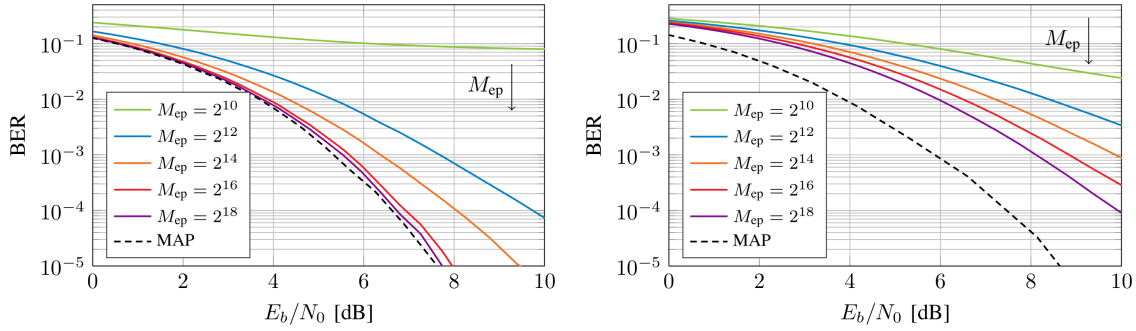
Uma listagem dos potenciais pontos de pesquisa e melhoria são listados na sequência.

1. Considerar outros modelos de aprendizado de máquina (ex.: máquinas de vetores de suporte, árvores de decisão, *boosting* etc.);
2. Empregar outros tipos de modulação (ex.: QAM, QPSK, FSK, etc.);
3. Modelar diferentes tipos de canais (foi adotado um com ruído AWGN);
4. Estimar a variância a partir da entrada e usar como parâmetro na camada LLR;
5. Realizar a otimização de hiperparâmetros;
6. Incorporar técnicas de regularização no treinamento da rede;
7. Alcançar desempenho MAP para códigos aleatórios;
8. Normalizar as entradas da rede neural para ter média zero e variância unitária;
9. Avaliar outras topologias de rede neural;
10. Melhorar o BLER para conjuntos de treinamento cada vez menores.

Considerando tais oportunidades de continuação do trabalho, o presente grupo propôs-se a desenvolver o item (5), considerando sua grande influência no desempenho de modelos de aprendizado de máquina.

7 Novos experimentos, resultados e conclusões

Levando em consideração tanto a codificação de código aleatório quanto polar, e empregando o otimizador Adam para minimização do erro quadrático médio (MSE – do inglês, *Mean Squared Error*), os autores alcançaram as taxas de erro de *bit* (BER – do inglês, *Bit Error Rate*) apresentadas na Figura 2, para quantidades de época de treinamento variando de 2^{10} a 2^{18} e um total de 800.000 *bits* simulados.



(a) Código polar

(b) Código aleatório

Figura 2: Influência do número de épocas M_{ep} na BER de uma RN 128-64-32 para um código de 16 bits de comprimento com taxa de código $r = 0.5$.

Na tentativa de obter resultados ainda melhores (neste sentido, BER com decaimento mais rápido e magnitude menor), propõe-se a avaliação de outros otimizadores, que serão comparados ao otimizador Adam [12] (empregado por Gruber et al. [2]), já que novas técnicas vem sendo propostas pela comunidade científica, tal como como o algoritmo Nadam [13].

Portanto, além do Adam, foram considerados os otimizadores brevemente apresentados a seguir[1] nos novos experimentos.

- SGD [15, 16]: uma das versões mais tradicionais do método gradiente descendente, realiza a atualização dos pesos imediatamente após a apresentação de cada exemplo de treinamento à rede neural.
- Adagrad [17]: adapta a taxa de aprendizado aos parâmetros, realizando atualizações menores (ou seja, baixas taxas de aprendizado) para parâmetros associados a características de ocorrência frequente e atualizações maiores (ou seja, altas taxas de aprendizado) para parâmetros associados a características não frequentes [14].
- Adadelta [18]: extensão do Adagrad que visa uma redução da taxa de aprendizagem por meio da restrição da janela de gradientes acumulados a um tamanho fixo [14].
- RMSprop [19]: contemporâneo ao Adadelta, é uma extensão do Adagrad que também visa lidar com suas taxas de aprendizagem radicalmente decrescentes. É idêntico ao Adadelta, mas que descarta de valores muito antigos do histórico de pesos, evitando que o modelo pare de aprender. [14]
- Adam (Adaptive Moment Estimation) [12]: além de armazenar uma média exponencialmente decrescente dos quadrados dos gradientes passados (como no Adadelta e no RMSprop), o método mantém uma média exponencialmente decrescente dos gradientes passados (como no *momentum* convencional). [14]
- AdaMax [12]: variação do Adam que visa obter uma convergência mais estável por meio do escalonamento do gradiente de forma inversamente proporcional à norma ℓ^∞ (em vez da ℓ^2) dos gradientes passados. [14]
- Nadam (Nesterov-accelerated Adaptive Moment Estimation) [13]: é uma combinação do Adam com o NAG (*Nesterov Accelerated Gradient* [20]) visando caminhar mais precisamente na direção do gradiente, atualizando os parâmetros com a etapa de *momentum* antes de calcular o gradiente. [14]

Os gráficos das Figuras 3 e 4 demonstram as BER para quantidades de épocas variando de 2^0 até 2^{21}, no caso dos código polar e aleatório, respectivamente.

No primeiro gráfico, que ilustra o experimento usando código polar, observa-se que o otimizador Nadam apresenta desempenho (em termos de taxa de erro de *bit*) superior ao Adam, utilizado no trabalho original de Gruber et al. [2]. Isto percebe-se pois sua curva BER decai mais rapidamente, aproximando-se rapidamente da curva MAP.

No segundo gráfico, que ilustra o experimento usando código aleatório, também é possível observar um resultado positivo no Nadam para uma quantidade de até 2^{20} épocas. Entretanto, a aproximação do decodificador neural ao MAP acontece de forma mais lenta, o que indica que a codificação aleatória é mais complexa e, consequentemente, seriam necessárias ainda mais épocas de treinamento.

Estes novos experimentos sugerem que é possível obter resultados melhores com decodificadores neurais mediante a otimização de hiperparâmetros. Neste caso, foram considerados outros otimizadores, mas há diversas possibilidades a serem exploradas. Além dos pontos levantados na seção 6, sugere-se o emprego de *early stopping* [21] para monitorar o decaimento do erro e evitar que a rede neural se sobreajuste.

[1] Uma comparação completa sobre eles pode ser encontrada no trabalho de Ruder [14], que aponta a evolução histórica destes e de outras versões do gradiente descendente, bem como semelhanças diferenças entre elas.

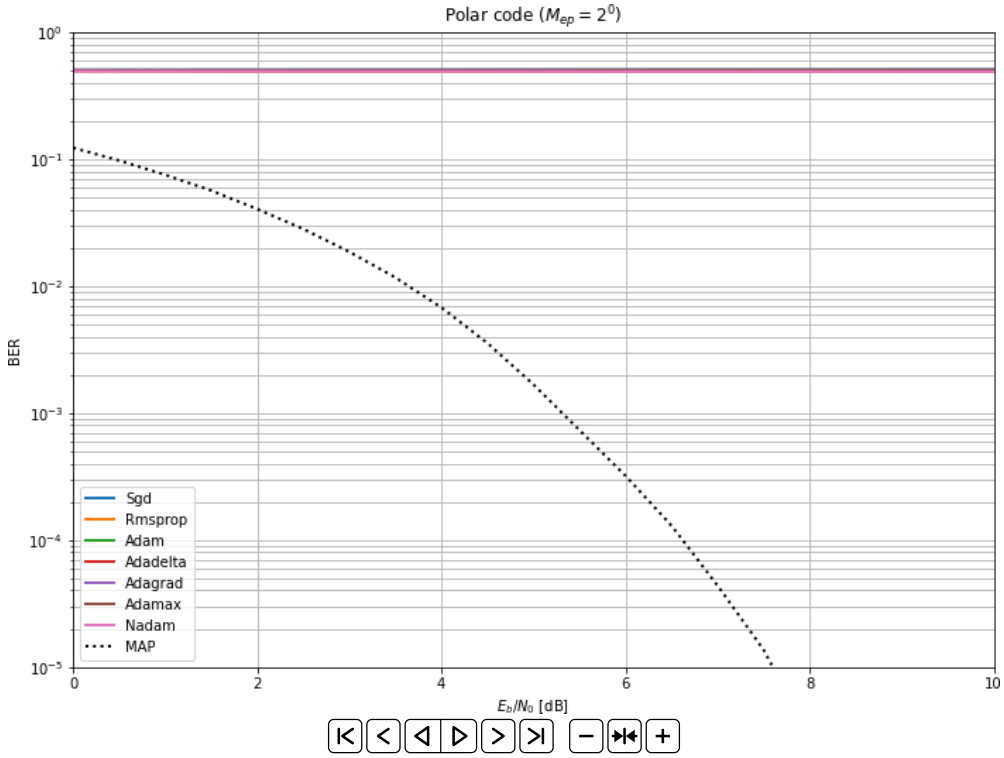


Figura 3: resultados da decodificação do código polar com $M_{ep} = 2^0, 2^1, \ldots, 2^{20}$ épocas.

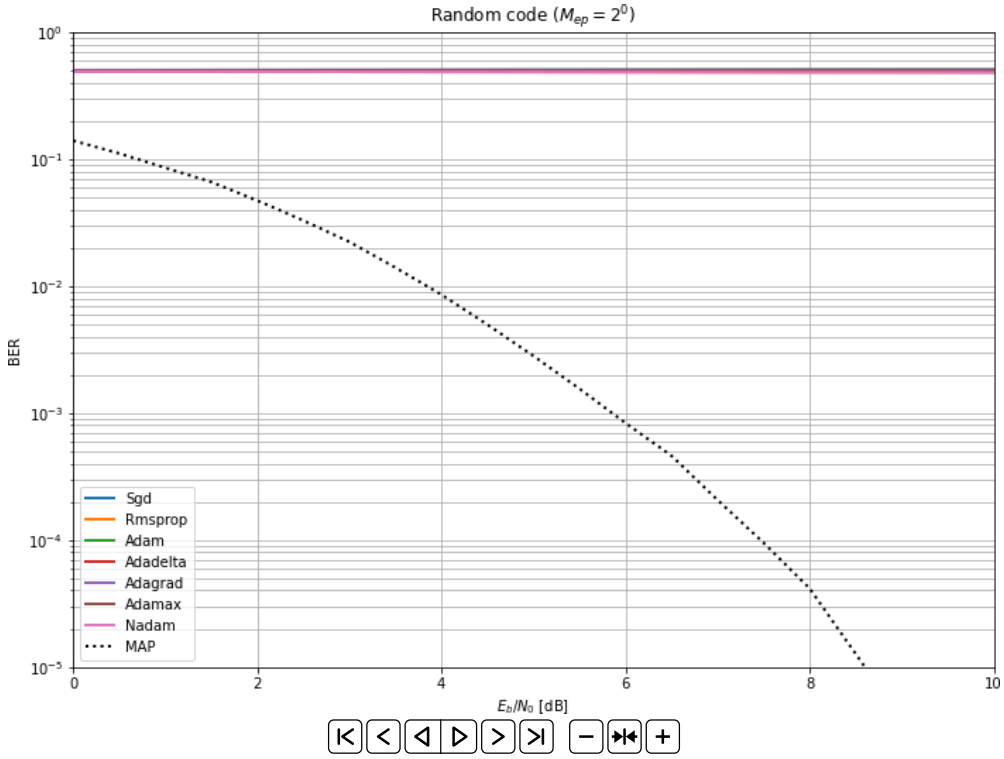


Figura 4: resultados da decodificação do código aleatório com $M_{ep} = 2^0, 2^1, \ldots, 2^{20}$ épocas.

Referências

[1] William R Caid e Robert W Means. "Neural network error correcting decoders for block and convolutional codes". Em: *[Proceedings] GLOBECOM'90: IEEE Global Telecommunications Conference and Exhibition*. IEEE. 1990, pp. 1028–1031.

[2] T. Gruber, S. Cammerer, J. Hoydis e S. t. Brink. "On deep learning-based channel decoding". Em: *2017 51st Annual Conference on Information Sciences and Systems (CISS)*. 2017, pp. 1–6. DOI: 10.1109/CISS.2017.7926071.

[3] 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.

[4] Gengsheng Zeng, Don Hush e Nasir Ahmed. "An application of neural net in decoding error-correcting codes". Em: *IEEE International Symposium on Circuits and Systems,* IEEE. 1989, pp. 782–785.

[5] LG Tallini e P Cull. "Neural nets for decoding error-correcting codes". Em: *IEEE Technical applications conference and workshops. Northcon/95. Conference record*. IEEE. 1995, p. 89.

[6] Ja-Ling Wu, Yuen-Hsien Tseng e Yuh-Ming Huang. "Neural network decoders for linear block codes". Em: *International Journal of Computational Engineering Science* 3.03 (2002), pp. 235–255.

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

[8] Hossam Abdelbaki, Erol Gelenbe e Said E El-Khamy. "Random neural network decoder for error correcting codes". Em: *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*. Vol. 5. IEEE. 1999, pp. 3241–3245.

[9] Eliya Nachmani, Yair Be'ery e David Burshtein. "Learning to decode linear codes using deep learning". Em: *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE. 2016, pp. 341–346.

[10] Wei Lyu, Zhaoyang Zhang, Chunxu Jiao, Kangjian Qin e Huazi Zhang. "Performance evaluation of channel decoding with deep neural networks". Em: *2018 IEEE International Conference on Communications (ICC)*. IEEE. 2018, pp. 1–6.

[11] Weihong Xu, Zhiwei Zhong, Yair Be'ery, Xiaohu You e Chuan Zhang. "Joint neural network equalizer and decoder". Em: *2018 15th International Symposium on Wireless Communication Systems (ISWCS)*. IEEE. 2018, pp. 1–5.

[12] Diederik P. Kingma e Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[13] Timothy Dozat. "Incorporating nesterov momentum into adam". Em: (2016).

[14] Sebastian Ruder. "An overview of gradient descent optimization algorithms". Em: *arXiv preprint arXiv:1609.04747* (2016).

[15] Herbert Robbins e Sutton Monro. "A stochastic approximation method". Em: *The annals of mathematical statistics* (1951), pp. 400–407.

[16] Jack Kiefer, Jacob Wolfowitz et al. "Stochastic estimation of the maximum of a regression function". Em: *The Annals of Mathematical Statistics* 23.3 (1952), pp. 462–466.

[17] John Duchi, Elad Hazan e Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." Em: *Journal of machine learning research* 12.7 (2011).

[18] Matthew D. Zeiler. *ADADELTA: An Adaptive Learning Rate Method*. 2012. arXiv: 1212.5701 [cs.LG].

[19] Geoffrey Hinton, Nitish Srivastava e Kevin Swersky. "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent". Em: *Cited on* 14.8 (2012).

[20] Yurii Nesterov. "A method for unconstrained convex minimization problem with the rate of convergence O (1/kˆ 2)". Em: *Doklady an ussr*. Vol. 269. 1983, pp. 543–547.

[21] Yuan Yao, Lorenzo Rosasco e Andrea Caponnetto. "On early stopping in gradient descent learning". Em: *Constructive Approximation* 26.2 (2007), pp. 289–315.