

# Decodificador baseado em Rede Neural Profunda para Códigos de Bloco Lineares Curtos Transmitidos via Canal Binário Simétrico

## Deep Neural Network-based Decoder for Short Linear Block Codes Transmitted via Binary Symmetric Channel

Jorge Kysnney Santos Kamassury

Universidade Federal de Santa Catarina (UFSC), Centro Tecnológico (CTC), Florianópolis, SC, Brasil

 <https://orcid.org/0000-0001-8335-9796>, [jorge.kamassury@posgrad.ufsc.br](mailto:jorge.kamassury@posgrad.ufsc.br).

---

### Informações do Artigo

#### Como citar este artigo

KAMASSURY, Jorge Kysnney Santos. Decodificador baseado em Rede Neural Profunda para Códigos de Bloco Lineares Curtos Transmitidos via Canal Binário Simétrico. **REMAT: Revista Eletrônica da Matemática**, Bento Gonçalves, RS, v. 7, n. 1, p. e3006, 22 fev. 2021. DOI: <https://doi.org/10.35819/remat2020v7i1id4389>



#### Histórico do Artigo

Submissão: 3 de agosto de 2020.

Aceite: 28 de setembro de 2020.

#### Palavras-chave

Decodificador Baseado em Rede Neural  
Canal Binário Simétrico  
Códigos Corretores de Erros

#### Keywords

Neural Network-Based Decoder  
Binary Symmetric Channel  
Error Correcting Codes

#### Resumo

Os códigos de comprimento curto têm sido alvo de estudos recentes devido, principalmente, às exigências de tecnologias emergentes por requisitos específicos de comunicação. Entretanto, para a classe de código mais promissora (BCH), a decodificação é complexa quando se usa os decodificadores tradicionais. Nesse contexto, os projetos que empregam redes neurais para esse propósito manifestam-se como interessantes alternativas. Isto posto, neste artigo estende-se, para os códigos BCH de comprimento  $n \leq 31$ , o projeto de decodificador proposto na literatura que aplica a rede neural para estimar o padrão de erro a partir da síndrome do vetor recebido. Além disso, introduz-se um novo decodificador que estima iterativamente as posições mais confiáveis para serem os bits errôneos do padrão de erro previamente predito por uma rede neural. Os resultados apresentados evidenciam que para todos os códigos analisados, o novo decodificador alcança os máximos desempenhos teóricos.

#### Abstract

Short-length codes have been the subject of recent studies mainly due to the need for specific communication requirements expressed by emerging technologies. However, for the most promising code class (BCH), decoding is complex when using traditional decoders. In this context, projects that use neural networks for this purpose appear as interesting alternatives. That said, in this article, the decoder project proposed in the literature that applies the neural network to estimate the error pattern considering the received vector syndrome extends to the BCH codes of length  $n \leq 31$ . In addition, a new decoder is introduced, one that iteratively estimates the most reliable positions to be the erroneous bits of the error pattern previously predicted by a neural network. The results presented show that, for all the analyzed codes, the new decoder reaches the maximum theoretical performances.

## 1 Introdução

Recentes investigações sobre projetos de códigos de comprimento curto têm adquirido notabilidade, especialmente, em razão das aplicações que emergentes tecnologias visam atender/viabilizar. A tecnologia 5G, por exemplo, entre outras categorias de serviços, almeja atender aplicações que demandam comunicação ultraconfiável (taxa de erro de bloco de  $10^{-3}$  a  $10^{-9}$ ) e de baixa latência ( $<100$  ms), como é o caso das redes elétricas inteligentes, telecirurgia, internet tátil e da automação fabril (SHIRVANIMOGHADDAM *et al.*, 2019). Porém, do ponto de vista da camada física, a comunicação sob essas condições é desafiadora, pois os próprios requisitos são estritos e conflitantes.

De acordo com Durisi, Koch e Popovski (2016), se por um lado, a minimização da latência pode ser obtida fazendo uso de pacotes curtos ou com a ampliação da largura de banda (embora essa última alternativa não seja sempre possível, especialmente para determinadas aplicações em controle industrial que podem operar sobre o espectro não-licenciado), por outro, o alcance do requisito de confiabilidade exige mais recursos (redundâncias e retransmissões) que acarretam o aumento da latência.

No contexto dos Códigos Corretores de Erros (*Error Correcting Codes*, ECCs), as pesquisas atuais têm sinalizado que a abordagem clássica que usa códigos de comprimento muito longos é inadequada em relação ao requisito da latência. Nesse sentido, estudos vêm sendo desenvolvidos para avaliar códigos curtos candidatos em relação a parâmetros como comprimento ( $k < 1000$  bits), confiabilidade, latência e complexidade algorítmica. Desses códigos (dentre os quais estão inclusos, códigos LDPC, códigos convolucionais, etc.), os códigos BCH estão entre aqueles que apresentam os resultados mais promissores. Contudo, um dos grandes desafios enfrentados pelos códigos BCH está relacionado ao fato de que sua decodificação, empregando os algoritmos tradicionais, é substancialmente complexa.

Uma alternativa interessante às técnicas tradicionais de decodificação consiste no uso de Redes Neurais (*Neural Networks*, NNs). De fato, ainda que não seja recente o emprego das NNs para esse propósito, o interesse por elas tem sido retomado, especialmente em razão do considerável potencial do Aprendizado Profundo (*Deep Learning*), incluindo a evolução da computação paralela e das GPUs (*Graphical Processing Units*).

Conforme demonstrado por Hornik, Stinchcombe e White (1989) e White (1990), as NNs do tipo *feedforward* permitem que a arquitetura aprenda qualquer mapeamento arbitrário; no caso

dos decodificadores, um mapeamento entre os padrões de entrada ruidosos e as palavras-código. Observa-se assim que nenhuma suposição sobre as estatísticas do ruído do canal de transmissão é necessária, pois a rede é teoricamente capaz de aprender o mapeamento ou extrair as características do canal a partir dos exemplos do conjunto de treinamento. Isto posto, pode-se supor que ao treinar uma NN usando todas as palavras-código possíveis de um determinado código, sejamos capazes de alcançar um desempenho de decodificação quase ótimo. Infelizmente, o processo de decodificação usando a NN está condenada pela *maldição da dimensionalidade* (WANG; WICKER, 1996); por exemplo, para um código de comprimento  $n = 100$  e uma taxa de  $r = 0.6$ , existem  $2^{60}$  palavras-código diferentes, o que facilmente estampa a inviabilidade desse procedimento de treinamento.

Sob essa perspectiva, Gruber, Cammerer, Hoydis e Brink (2017) reconhecem que o único modo pelo qual uma NN possa ser treinada para decodificar códigos com comprimentos de interesse prático é se ela pode estender o aprendizado obtido com uma fração do *codebook*<sup>1</sup>; em outras palavras, a NN deve ser capaz de aprender conceitos de decodificação que se aplicam a todas as palavras-código. Uma interessante forma de incorporar o conhecimento do código no projeto de decodificação usando NN foi apresentada por Tallini e Cull (1995), em que uma NN com duas camadas ocultas é alimentada pelo vetor síndrome de uma palavra-código transmitida via canal binário simétrico (*Binary Symmetric Channel*, BSC) e produz a estimativa do padrão de erro. Embora não alcancem os desempenhos ótimos de decodificação, Tallini e Cull (1995) obtêm razoáveis performances para o código de Hamming (7,4) e o código de Reed-Muller de 2ª ordem (32,16).

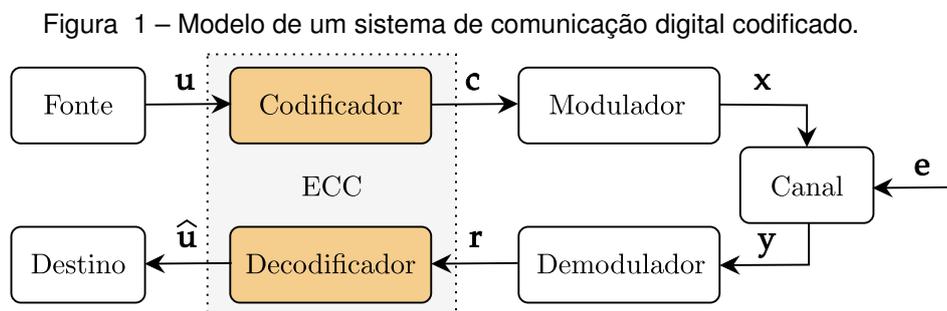
Diante do exposto, no presente artigo estende-se a abordagem apresentada por Tallini e Cull (1995) para os códigos BCH com  $n \leq 31$  usando uma nova arquitetura neural profunda. Além disso, introduz-se um novo algoritmo de decodificação intitulado *Neural-Max* (NM), também alimentado pelo vetor síndrome, que atua estimando iterativamente as posições mais confiáveis a serem os bits errôneos do padrão de erro, por sua vez, predito por uma NN. Os resultados evidenciam que o decodificador NM tem desempenho superior ao decodificador proposto por Tallini e Cull (1995).

---

<sup>1</sup>Conjunto de todas as palavras-código de um determinado código.

## 2 Aspectos gerais sobre códigos de blocos

De início, considere um modelo de sistema de comunicação digital codificado ilustrado na Figura 1. Nesse esquema, uma sequência de dígitos  $q$ -ária gerada por uma fonte é subdividida em blocos de  $k$  dígitos cada, denotados pelo vetor  $\mathbf{u} = [u_0, u_1, \dots, u_{k-1}]$  que, por sua vez, alimentam o codificador; a ação do codificador resume-se, sob certas regras, em adicionar redundância ao vetor  $\mathbf{u}$ , produzindo um vetor  $\mathbf{c} = [c_0, c_1, \dots, c_{n-1}]$  denominado de *palavra-código*, onde  $n \geq k$ . Para o conjunto ( $q^k$ ) de todas as palavras-código atribui-se o nome de *código de bloco*.



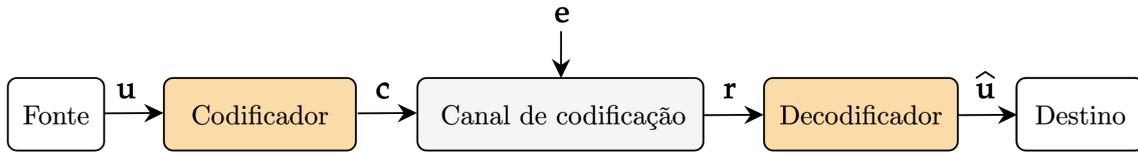
Fonte: Elaboração do autor (2020).

Após a geração das palavras-código, o bloco modulador transforma o vetor codificado  $\mathbf{c}$  em um vetor de sinal modulado  $\mathbf{x}$  apropriado para transmissão usando canais analógicos (links de satélites, linhas telefônicas, etc.). Como os canais estão sujeitos às várias fontes de erros  $\mathbf{e}$  (ruídos aleatórios, interferências e falhas), a saída do canal  $\mathbf{y}$  pode ser diferente do vetor de entrada  $\mathbf{x}$ .

No lado do receptor, o bloco demodulador executa uma operação inversa e produz uma saída  $\mathbf{r} = [r_0, r_1, \dots, r_{n-1}]$ . Em razão dos possíveis erros durante a transmissão, o bloco decodificador faz uso da redundância no vetor  $\mathbf{c}$  para corrigir esses erros no vetor recebido  $\mathbf{y}$  e assim obter uma estimativa  $\hat{\mathbf{u}} = [\hat{u}_0, \hat{u}_1, \dots, \hat{u}_{k-1}]$ .

Para efeitos de simplificação, os detalhes das seções seguintes estão baseados no esquema da Figura 2, onde os blocos modulador, canal e demodulador estão combinados em um único bloco intitulado *canal de codificação*, para o qual, a entrada e saída são denotadas por  $\mathbf{c}$  e  $\mathbf{r}$ , nessa ordem (LIN; COSTELLO, 2004). Ademais, lida-se apenas com códigos de bloco lineares binários ( $q = 2$ ).

Figura 2 – Modelo simplificado de um sistema de comunicação digital codificado.



Fonte: Elaboração do autor (2020).

## 2.1 Códigos de bloco lineares e descrição matricial

Um código de bloco de comprimento  $n$  e  $2^k$  palavras-código é dito um *código linear*  $(n, k)$  se, e somente se, suas  $2^k$  palavras-código formam um subespaço  $k$ -dimensional do espaço vetorial de todas as  $n$ -tuplas sobre  $GF(2)$ . Nesse caso, em virtude da linearidade do código, é possível selecionar  $k$  palavras-código linearmente independentes formando uma base de um espaço linear  $k$ -dimensional de modo que qualquer palavra-código possa ser gerada por uma combinação linear das  $k$  palavras-código da base (SANTOS KAMASSURY; TÔRRES; DUARTE, 2019). Essas  $k$  palavras-código podem ser obtidas fazendo uso da *matriz geradora* denotada por  $\mathbf{G}$ . Desse modo, para  $k$  e  $n$  quaisquer, tem-se:

$$\mathbf{c} = u_0\mathbf{G}_0 + u_1\mathbf{G}_1 + u_2\mathbf{G}_2 + \dots + u_{k-1}\mathbf{G}_{k-1} = \mathbf{u}\mathbf{G} \quad (1)$$

Pode-se ainda incluir os dígitos de informação inalterados na palavra-código. Um código de bloco com essa característica é chamado de *sistemático* (PROAKIS; SALEHI, 2007). Nesse caso, a palavra-código é denotada por

$$\mathbf{c} = [c_0, c_1, \dots, c_{n-k-1}, u_0, u_1, \dots, u_{k-1}] \quad (2)$$

e a matriz  $\mathbf{G}$  possuirá a forma

$$\mathbf{G}_{k \times n} = [\mathbf{P}_{k \times n-k} | \mathbf{I}_k] \quad (3)$$

onde  $\mathbf{I}$  é a matriz identidade de ordem  $k$  e  $\mathbf{P}$  é a *matriz de paridade* e representa os dígitos utilizados para detecção e/ou correção de erro.

Da teoria de espaços lineares, é conhecida ainda a existência de uma matriz  $\mathbf{H}$  com dimensão  $(n - k) \times n$  cujas  $n - k$  linhas são linearmente independentes, tal que:

$$\mathbf{G}\mathbf{H}^T = 0, \quad \mathbf{c}\mathbf{H}^T = 0 \quad (4)$$

Segue a partir da Equação 4 que qualquer palavra-código  $\mathbf{c}$  gerada por  $\mathbf{G}$  é ortogonal às linhas de  $\mathbf{H}$ ; conclui-se assim que uma  $n$ -upla  $\mathbf{c}$  é uma palavra-código de  $\mathcal{C}$  se, e somente se, respeitar a Equação 4, onde  $\mathbf{H}$  é conhecida como *matriz de verificação de paridade* do código linear  $\mathcal{C}$ , sendo denotada (na forma sistemática) por:

$$\mathbf{H}_{n-k \times n} = [\mathbf{I}_{n-k} | \mathbf{P}^T] \quad (5)$$

De modo geral, a Equação 1 (geradora/codificadora) e a Equação 4 (verificação de paridade) representam a base para a descrição operacional dos códigos de bloco lineares. Resumidamente, enquanto no transmissor a matriz  $\mathbf{G}$  é usada na codificação, a matriz  $\mathbf{H}$  é aplicada no receptor para o processo de decodificação.

## 2.2 Síndrome

Considere uma palavra-código  $\mathbf{c}$  transmitida por um canal ruidoso, cujo vetor recebido  $\mathbf{r}$  seja expresso por

$$\mathbf{r} = \mathbf{c} + \mathbf{e} \quad (6)$$

onde  $\mathbf{e}$  é denominado de *padrão de erro* ou *vetor de erro* (HAYKIN, 2014). Conforme discutido na seção 1, a tarefa do decodificador será estimar  $\hat{\mathbf{u}}$  a partir de  $\mathbf{r}$ . Comumente, inicia-se o algoritmo (usado para decodificação) computando o vetor denominado *síndrome* que é definido por

$$\mathbf{s} = \mathbf{r}\mathbf{H}^T \quad (7)$$

e possui as seguintes propriedades (HAYKIN, 2014):

- i) **Propriedade 1:** o vetor síndrome depende unicamente do padrão de erro e não da palavra-código transmitida.

$$\mathbf{s} = \mathbf{e}\mathbf{H}^T \quad (8)$$

- ii) **Propriedade 2:** Todos os padrões de erro que diferem por uma palavra-código têm a mesma síndrome.

De forma resumida, considere  $k$  dígitos de informação. Nesse caso, há  $2^k$  palavras-código distintas denotadas por  $\mathbf{c}_i$  onde  $0 \leq i \leq 2^k - 1$ . Analogamente, para qualquer  $\mathbf{e}$ , define-se  $2^k$  vetores distintos  $\mathbf{e}_i$  como se segue:

$$\mathbf{e}_i = \mathbf{e} + \mathbf{c}_i \quad (9)$$

O conjunto de  $\mathbf{e}_i$  é chamado de *coset* do código  $\mathcal{C}$  e possui  $2^k$  elementos que diferem no máximo por uma palavra-código. Pode-se verificar que a relação  $\mathbf{e}_i \mathbf{H}^T = \mathbf{e} \mathbf{H}^T$  é independente do índice  $i$ , do qual se conclui que cada *coset* é caracterizado por uma única síndrome.

De fato, ambas as propriedades da síndrome podem ser analisadas a partir da Equação 8. Sob essa perspectiva e usando a forma sistemática da matriz  $\mathbf{H}$ , verifica-se que os  $n - k$  elementos da síndrome são combinações lineares dos  $n$  elementos do padrão de erro  $\mathbf{e}$ . O conjunto dessas  $n - k$  equações evidenciam que a síndrome contém informações sobre os padrões de erros  $\mathbf{e}$ , portanto, pode ser usado para detecção de erros.

### 2.2.1 Decodificação via síndrome

Seja o vetor recebido  $\mathbf{c}$  com  $2^n$  valores possíveis, a ação do decodificador implica em particionar esse conjunto  $2^n$  em  $2^k$  subconjuntos disjuntos  $\Omega_1, \dots, \Omega_{2^k}$  de tal forma que  $\Omega_i$  corresponda à  $\mathbf{c}_i$  para  $1 \leq i \leq 2^k$ . Com efeito, para que a decodificação seja bem-sucedida,  $\mathbf{r}$  deve estar no subconjunto pertencente a  $\mathbf{c}_i$  que realmente foi enviada. Os conjuntos  $2^k$  constituem o *arranjo padrão* de um código de bloco linear (vide Figura 3), cujas etapas para sua construção podem ser consultadas em Lin e Costello (2004).

Figura 3 – Arranjo padrão para um código linear  $\mathcal{C}(n, k)$ .

$\mathbf{c}_1 = 0$	$\mathbf{c}_2$	$\mathbf{c}_3$	...	$\mathbf{c}_i$	...	$\mathbf{c}_{2^k}$
$\mathbf{e}_2$	$\mathbf{c}_2 + \mathbf{e}_2$	$\mathbf{c}_3 + \mathbf{e}_2$	...	$\mathbf{c}_i + \mathbf{e}_2$	...	$\mathbf{c}_{2^k} + \mathbf{e}_2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$
$\mathbf{e}_j$	$\mathbf{c}_2 + \mathbf{e}_j$	$\mathbf{c}_3 + \mathbf{e}_j$	...	$\mathbf{c}_i + \mathbf{e}_j$	...	$\mathbf{c}_{2^k} + \mathbf{e}_j$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$
$\mathbf{e}_{2^{n-k}}$	$\mathbf{c}_2 + \mathbf{e}_{2^{n-k}}$	$\mathbf{c}_3 + \mathbf{e}_{2^{n-k}}$	...	$\mathbf{c}_i + \mathbf{e}_{2^{n-k}}$	...	$\mathbf{c}_{2^k} + \mathbf{e}_{2^{n-k}}$

Fonte: Elaboração do autor (2020).

As  $2^k$  colunas dessa matriz representam os subconjuntos disjuntos, enquanto as  $2^{n-k}$  linhas consistem nos *cosets* do código, sendo seus primeiros elementos  $\mathbf{e}_2, \mathbf{e}_3, \dots, \mathbf{e}_{2^{n-k}}$  denominados de *líderes de cosets*.

Na prática, para um certo canal de transmissão, a probabilidade de erro de decodificação é reduzida quando os padrões de erro mais prováveis são selecionados como líderes de *cosets*. No

caso de uma transmissão via BSC, quanto menor o peso de Hamming<sup>2</sup>  $w$ , maior a probabilidade de ocorrência de erro (HAYKIN, 2014). Por consequência, o arranjo padrão é construído tendo como líderes de *cosets* os vetores com os menores pesos de Hamming.

De forma abreviada, o processo de decodificação possui as seguintes etapas:

- i) De posse do vetor recebido  $\mathbf{r}$ , computa-se a síndrome:  $\mathbf{s} = \mathbf{r}\mathbf{H}^T$ ;
- ii) Se  $\mathbf{s} = \mathbf{0}$ , considera-se que  $\mathbf{r}$  está livre de erro. Caso contrário, a partir do arranjo padrão, avalia-se o *coset* caracterizado por  $\mathbf{s}$ , identificando o *coset* líder (padrão de erro com maior probabilidade de ocorrência) e denotando-o por  $\mathbf{e}_0$ ;
- iii) Estima-se  $\hat{\mathbf{c}} = \mathbf{r} + \mathbf{e}_0$  como a versão decodificada de  $\mathbf{r}$ .

### 3 Códigos BCH

Um das classes mais importantes e poderosas de códigos de bloco lineares são os códigos BCH. Descobertos por Bose, Chaudhuri e Hocquenghem, são códigos cíclicos com uma ampla variedade de parâmetros (HOCQUENGHEM, 1959, BOSE; RAY-CHAUDHURI, 1960a, 1960b). Na prática, os códigos BCH binários *primitivos* são os mais comuns, sendo caracterizados pelos parâmetros inteiros e positivos  $m$  e  $t$  que respeitam as seguintes condições:

- Comprimento do bloco:  $n = 2^m - 1$ ;
- Quantidade de bits de mensagem:  $k \geq n - mt$ ;
- Distância mínima<sup>3</sup>:  $d_{\min} \geq 2t + 1$

Esses códigos, em específico, possuem capacidade de correção de  $t$  erros, ou seja, podem detectar e corrigir múltiplos erros, sendo assim considerados uma generalização dos códigos de Hamming. Além disso, possuem flexibilidade na escolha de parâmetros do código como comprimento e taxa de código (PROAKIS; SALEHI, 2007). Essas características tornam os códigos BCH extremamente atrativos, especialmente, para cenários que demandam códigos de comprimento curto.

---

<sup>2</sup>Refere-se ao número de elementos não-nulos de um vetor.

<sup>3</sup>Em teoria da informação, a distância de Hamming ( $d$ ) é compreendida como o número de posições em que dois vetores (no nosso caso, palavras-código) diferem. A distância mínima ( $d_{\min}$ ), por sua vez, é a menor distância de Hamming entre duas palavras-códigos válidas de um código.

## 4 Redes Neurais Profundas

Em termos gerais, as Redes Neurais Profundas (*Deep Neural Networks*, DNNs) consistem em aproximações de funções compostas por uma série de camadas, em que cada camada representa alguma transformação de ativações de entrada para saída com base em uma função de transferência paramétrica com um determinado conjunto de pesos aprendidos (O'SHEA; HOYDIS, 2017).

Genericamente, uma DNN do tipo *perceptron* com  $L$  camadas representa um mapeamento  $f(\mathbf{r}_0; \theta) : \mathbb{R}^{N_0} \mapsto \mathbb{R}^{N_L}$  de um vetor de entrada  $\mathbf{r}_0 \in \mathbb{R}^{N_0}$  para um vetor de saída  $\mathbf{r}_L \in \mathbb{R}^{N_L}$  por meio de  $L$  etapas iterativas, tal que:

$$\mathbf{r}_l = f_l(\mathbf{r}_{l-1}; \theta_l), \quad l = 1, \dots, L \quad (10)$$

Para o caso de uma arquitetura com camadas totalmente conectadas, a  $l$ -ésima camada (vide Figura 4) tem o mapeamento descrito pela seguinte forma

$$f_l(\mathbf{r}_{l-1}; \theta_l) = g(\mathbf{W}_l \mathbf{r}_{l-1} + \mathbf{b}_l) \quad (11)$$

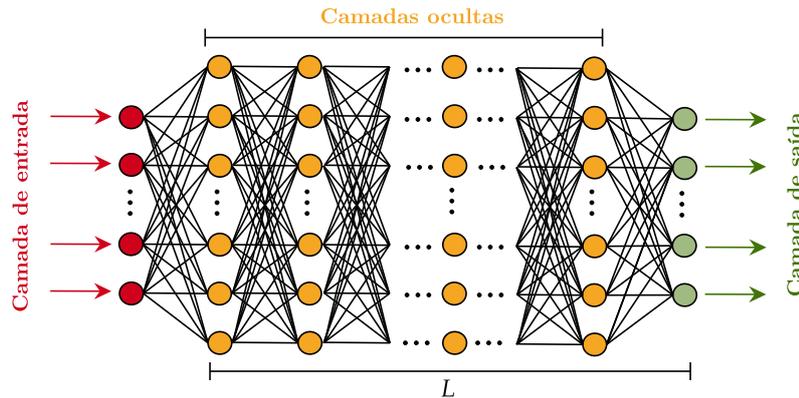
onde:

- $\mathbf{W}_l \in \mathbb{R}^{N_l \times N_{l-1}}$  é a matriz de pesos da respectiva camada  $l$ ;
- $\mathbf{b}_l \in \mathbb{R}^{N_l}$  denota o vetor de *bias*;
- $g(\cdot)$  representa a *função de ativação*, em geral, aplicada individualmente a cada elemento de entrada da camada, isto é  $[g(\mathbf{z})]_i = g(z)_i$ ;
- $\theta_l = \{\mathbf{W}_l, \mathbf{b}_l\}$  é o conjunto de parâmetros da camada  $l$ .

Com efeito, a principal característica da arquitetura exposta na Figura 4 reside no fato de que cada neurônio recebe na entrada a contribuição de cada neurônio da camada anterior e alimenta sua saída para todos os neurônios da próxima camada. Vale mencionar que a função de ativação  $g(\cdot)$  presente na Equação 11, ao introduzir uma não-linearidade, contribui significativamente para a capacidade de aprendizagem/mapeamento da rede. De fato, na ausência da não-linearidade, o empilhamento de camadas — próprio das redes profundas — não resultaria em grande vantagem (GOODFELLOW; BENGIO; COURVILLE, 2016).

No contexto do aprendizado de máquina, reconhece-se que o poder das NNs reside intimamente na sua configurabilidade. Normalmente, a rede é configurada por um determinado procedimento de treinamento que depende de um conjunto de dados de amostra, consistindo de entradas

Figura 4 – Exemplo de arquitetura DNN do tipo *perceptron* com camadas de entrada, ocultas e de saída.



Fonte: Elaboração do autor (2020).

e, no caso de treinamento supervisionado, de saídas desejadas (conhecidas como *rótulos*). Desse modo, o processo de treinamento de uma NN é realizado usando dados rotulados, isto é, um conjunto de pares de vetores entrada-saída  $(\mathbf{r}_{0,j}, \mathbf{r}^\dagger_{L,j})$  para  $j \in [1, \dots, N]$ , onde  $\mathbf{r}^\dagger_{L,j}$  é o valor da saída desejada. Nesse sentido, o objetivo do treinamento incorre em minimizar a *função de custo*

$$J(\theta) = \frac{1}{N} \sum_{j=1}^N \mathcal{L}(\mathbf{r}^\dagger_{L,j}, \mathbf{r}_{L,j}) \quad (12)$$

em relação ao conjunto de parâmetros  $\theta$  em que  $\mathcal{L} : \mathbb{R}^{N_L} \times \mathbb{R}^{N_L} \mapsto \mathbb{R}$  corresponde à *função de perda*. Em outras palavras, conforme enfatizam O’shea e Hoydis (2017), esse processo de aprendizagem visa a obtenção de bons conjuntos de parâmetros  $\theta$  que minimizam a Equação 12. Um dos mais populares algoritmos para encontrar esses conjuntos é o gradiente descendente estocástico (*stochastic gradient descent*, SGD) combinado com o método de *backpropagation*, no qual, para um valor inicial aleatório  $\theta = \theta_0$ , a atualização iterativa dos parâmetros  $\theta$  é denota por

$$\theta_{t+1} = \theta_t - \alpha \nabla \hat{J}(\theta_t) \quad (13)$$

onde  $\alpha$  é o parâmetro que define a taxa de aprendizado (*learning rate*, lr), isto é, o tamanho do passo que será dado na direção apontada pelo gradiente que, por sua vez, é calculado através do algoritmo de *backpropagation* (BISHOP, 2006). Na Equação 13,  $\hat{J}(\theta)$  consiste na aproximação da função custo computada para exemplos aleatórios de treinamento (*mini-batches*)  $\Gamma_t \subseteq \{1, \dots, N\}$  de tamanho fixo  $\Gamma_t$  para cada iteração, isto é:

$$\hat{J}(\theta) = \frac{1}{\Gamma_t} \sum_{j \in \Gamma_t} \mathcal{L}(\mathbf{r}^\dagger_{L,j}, \mathbf{r}_{L,j}) \quad (14)$$

Na prática, devido ao fator aleatório inerente, o SGD fornece uma aproximação grosseira do gradiente descendente, resultando em uma convergência não suave. Ademais, para conjuntos de dados grandes, o SGD é custoso computacionalmente em razão das constantes atualizações exigidas por este. Por esses motivos, outras variantes como RMSProp, AdaGrad, AdaDelta e Adam são mais utilizadas (GOODFELLOW; BENGIO; COURVILLE, 2016).

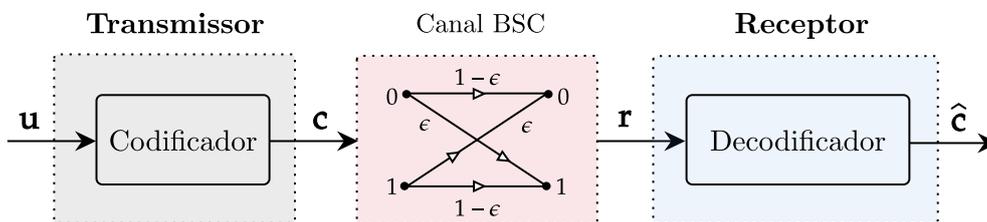
## 5 Decodificador projetado por Tallini e Cull

De início, seja um sistema de comunicação cuja transmissão utiliza um código linear  $\mathcal{C}$  com taxa de codificação  $\frac{k}{n}$ , em que uma mensagem original de  $k$  bits denotada por  $\mathbf{u} \in \text{GF}(2)^k$  é codificada para uma palavra-código  $\mathbf{c}$  de  $n$  bits, através da relação

$$\mathbf{c} = \mathbf{u}\mathbf{G}, \quad \mathbf{c} \in \text{GF}(2)^n \tag{15}$$

onde  $\mathbf{G} \in \text{GF}(2)^{n \times k}$  é a matriz geradora. A Figura 5 apresenta o modelo simplificado de um sistema de comunicação que transmite via BSC com probabilidade de erro de transição  $\epsilon$ .

Figura 5 – Modelo simplificado de um sistema de comunicação usando BSC.



Fonte: Elaboração do autor (2020).

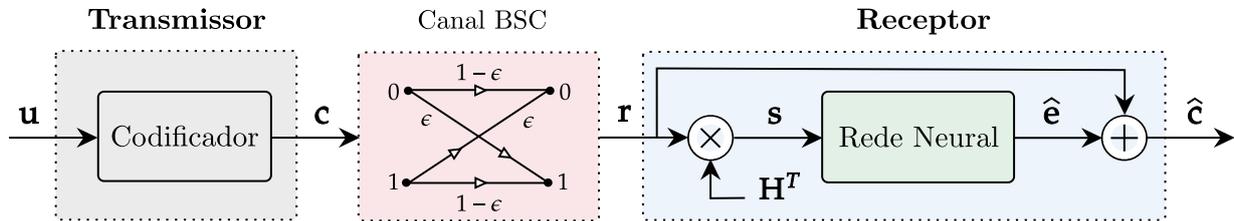
Para esse canal de transmissão, uma estratégia de decodificação que usa a síndrome ( $\mathbf{s}$ ) no treinamento de uma NN para estimar os padrões de erro  $\hat{\mathbf{e}}$  foi originalmente apresentada por Tallini e Cull (1995). Ilustrado na Figura 6, o esquema proposto pode ser resumido em três etapas, a saber:

- i) Para uma palavra ruidosa  $\mathbf{r}$  computa-se a síndrome  $\mathbf{s} = \mathbf{r}\mathbf{H}^T$ ;
- ii) De posse do vetor síndrome, estima-se através de uma NN o padrão de erro mais provável ( $\hat{\mathbf{e}}$ ), ou seja, aquele com o peso mínimo;
- iii) Computa-se a estimativa  $\hat{\mathbf{c}}$  da palavra-código enviada  $\mathbf{c}$ , isto é <sup>4</sup>:

$$\hat{\mathbf{c}} = \mathbf{r} + \hat{\mathbf{e}} \tag{16}$$

<sup>4</sup>A operação de adição  $\mathbf{r} + \hat{\mathbf{e}}$  é em módulo-2.

Figura 6 – Decodificador baseado em síndrome proposto por Tallini e Cull (1995) para o BSC.



Fonte: Elaboração do autor (2020).

Endossando o que foi mencionado na seção 1, usando esse decodificador, Tallini e Cull (1995) obtiveram bons resultados ao lidarem com códigos curtos como Hamming(7,4) e Reed-Muller de 2ª ordem (32,16). Na seção 8, utiliza-se essa mesma estrutura de decodificação, porém, visando exclusivamente códigos BCH curtos de comprimento  $n \leq 31$ .

## 6 Decodificador Neural-Max

No decodificador proposto por Tallini e Cull (1995), presume-se que a NN seja capaz de estimar o vetor de erro por completo, ou seja, todas as possíveis posições dos erros. Contudo, o sucesso dessa estimativa pode ser comprometido por padrões de erros que promovam inconsistências durante o processo de treinamento.

Para uma compreensão efetiva dessas inconsistências, suponha, por exemplo, uma palavra-código  $c' \in \mathcal{C}$  dada por:

$$c' = [0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0]$$

Nesse caso, é fácil verificar que dois possíveis padrões de erros tal que  $c' = e_1 + e_2$  são

$$e_1 = [0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$$

$$e_2 = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0]$$

com síndromes  $s_1 = s_2$ , onde  $w(e_1) < w(e_2)$ . Com base no esquema da Figura 6, no estágio de treinamento, a NN deve aprender a estimar o padrão de erro com base nos exemplos observados; assim, para uma síndrome  $s_1$ , a rede é treinada para estimar  $\hat{e}_1$ , enquanto que para a síndrome

$s_2$ , a rede deve prever  $\hat{e}_2$ . Com efeito, a inconsistência ocorre pelo fato de que para essas duas entradas iguais ( $s_1$  e  $s_2$ ), exigimos que a rede decida corretamente entre  $\hat{e}_1$  e  $\hat{e}_2$ .

Sob essa perspectiva, diante das síndromes ( $s_1$  e  $s_2$ ) e seus correspondentes padrões de erro ( $e_1$  e  $e_2$ ), os resultados evidenciam que a NN, na etapa de predição, decide mais vez por  $e_1$ . Isso ocorre porque, por possuir  $w$  menor, os exemplos de  $e_1$  são mais prováveis de ocorrerem. Convém enfatizar que isso não indica que predições por  $e_2$  sejam inviáveis, e sim, que são menos frequentes.

Por outro lado, sejam os seguintes exemplos de treinamento ( $s_3, e_3$ ) e ( $s_4, e_4$ ), onde  $s_3 = s_4$  e

$$e_3 = [0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0]$$

$$e_4 = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0].$$

Observa-se que, por possuírem pesos iguais,  $e_3$  e  $e_4$  contribuem igualmente durante o treinamento da rede. Nesse caso, como os exemplos  $e_3$  e  $e_4$  ocorrem com a mesma frequência, a predição da rede tende a falhar ainda mais em decidir qual o padrão de erro mais provável entre  $e_3$  e  $e_4$  para uma síndrome  $s = s_3 = s_4$ .

Reconhecendo essa inconsistência e, por consequência, a falha da predição da rede, propõe-se um algoritmo de decodificação intitulado *Neural-Max* (NM) no qual a NN só precisa estimar com confiança elevada uma única posição do padrão de erro  $\hat{e}$  para cada iteração. É relevante ressaltar que essa é uma abordagem diferente daquela proposta por Tallini e Cull (1995), na qual a NN estima ao mesmo tempo todas as posições do vetor de erro. De fato, a estimativa iterativa do decodificador NM pode ser compreendida como um pós-processamento otimizado da saída obtida com decodificador projetado por Tallini e Cull (1995).

As etapas para a implementação desse novo decodificador (para o BSC) estão sumarizadas no Quadro 1. Observe que para a inicialização do decodificador NM são necessárias as seguintes entradas: a síndrome  $s$  previamente calculada, o comprimento da palavra-código  $n$  e a matriz  $\mathbf{H}$ . Se  $s \neq 0$ , decide-se logo pelo padrão de erro nulo; caso contrário, seguem as etapas elencadas no Quadro 1. Devido ao fato de que não há garantia que sempre seja encontrada uma síndrome nula, define-se um valor máximo ( $n$ ) de iterações para cada vetor  $\mathbf{r}$  recebido pelo decodificador.

## 7 Arquitetura neural

Para a aplicação dos decodificadores discutidos nas seções 5 e 6, propõe-se a DNN da Figura 7, constituída de cinco camadas (quatro ocultas e uma de saída), na qual as quatro primeiras camadas possuem  $8n$  neurônios cada, enquanto a última camada (saída) possui  $n$  neurônios, sendo  $n$  o comprimento da palavra-código do código em questão. Na figura 7, o símbolo  $||$  representa a operação de concatenação.

### Quadro 1: Algoritmo do decodificador Neural-Max

**Entrada:**  $r, n, H$

**Saída:**  $\hat{e}$

1 **início**

2 - Inicia-se um vetor de padrão de erro  $\hat{e}$  como um vetor nulo

3 - Inicia-se um contador  $j$  para limitar a quantidade de iterações

4 - Calcula-se a síndrome  $s$  do vetor recebido  $r$

5 **enquanto**  $s \neq 0$  e  $j < n$  **faça**

6 - Prediz-se o padrão de erro  $\hat{e}_{est}$  a partir da rede neural treinada

7 - Cria-se um vetor  $\hat{e}_{max}$  com valor 1 na posição que corresponda à posição que possui o maior valor de  $\hat{e}_{est}$ . Para as demais posições, os valores serão nulos

8 - Computa-se a síndrome  $\hat{s}$  a partir de  $\hat{e}_{max}$

9 - Atualiza-se a síndrome  $s$  pela soma em módulo-2 entre  $s$  e  $\hat{s}$

10 - Atualiza-se o padrão de erro  $\hat{e}$  pela soma em módulo-2 entre  $\hat{e}$  e  $\hat{e}_{max}$

11 - Incrementa-se o contador  $j$

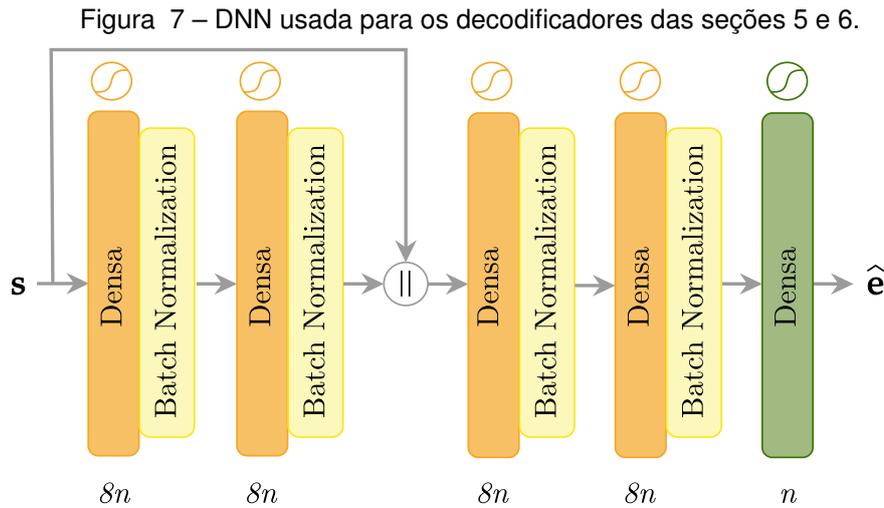
12 **fim**

13 **fim**

Fonte: Elaboração do autor (2020).

Inspirado na abordagem usada por Kamassury e Silva (2020), introduz-se uma camada de normalização em lote (*Batch Normalization*, BN) após cada camada oculta; essas camadas BN são empregadas para estabilizar e acelerar o processo de treinamento da DNN (IOFFE; SZEGEDY, 2015).

Por fim, conforme exigem ambos os decodificadores, a DNN da Figura 7 é alimentada pela síndrome do vetor  $r$  e estima o padrão de erro  $\hat{e}$ . Em todas as camadas, usamos a função de ativação *sigmóide* para os neurônios e a distribuição normal para a inicialização dos parâmetros da rede.



Fonte: Elaboração do autor (2020).

## 8 Resultados das simulações e discussões

Nessa seção, apresentam-se os resultados das simulações para o decodificador projetado por Tallini e Cull (1995) e o decodificador NM. Para as etapas de treinamento e teste da arquitetura neural proposta (Figura 7) utilizou-se a biblioteca Keras com Tensorflow como *backend*. Todas as implementações foram feitas na plataforma colab (*google colab*<sup>5</sup>).

É relevante acentuar que nas simulações dessa seção, não foram usados conjuntos de treinamento fixos dos códigos analisados, pois dependendo do código, até frações dos seus correspondentes *codebooks* podem ser extremamente onerosos computacionalmente no que se refere à geração e carregamento (na memória). Além do mais, no treinamento de conjuntos fixos, os mesmos exemplos são utilizados várias vezes, o que pode resultar em *overfitting*.

Para contornar esses problemas, gerou-se em tempo real os exemplos, utilizando-os imediatamente no treinamento do modelo da rede. Com essa abordagem, como os *batches* são independentes e não é necessário armazenar os exemplos já utilizados no treinamento, consegue-se evitar simultaneamente o *overfitting* e o problema com espaço de memória. Isto posto, para o treinamento da DNN com uma quantidade específica de exemplos, basta a alteração da quantidade de *época*<sup>6</sup> e do tamanho do *batch*.

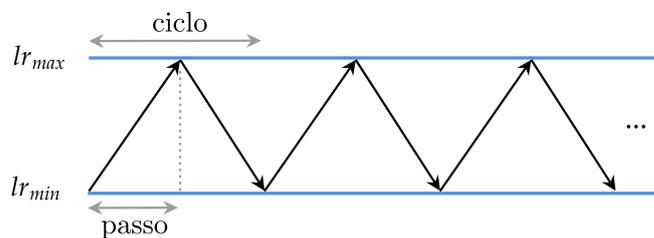
<sup>5</sup><https://colab.research.google.com/>

<sup>6</sup>Define-se época como a quantidade de passagens completas pelo conjunto de treinamento. Para conjuntos não fixos, nos quais os exemplos de treinamento são gerados em tempo real, o hiperparâmetro épocas corresponde à quantidade de vezes que se deseja gerar exemplos e usá-los no treinamento

## 8.1 Treinamento

No estágio de treinamento da rede neural, utilizou-se o algoritmo *Adam* (KINGMA; BA, 2014) para a otimização dos pesos da rede e a *entropia cruzada binária* como função de perda. Definiu-se ainda o tamanho do *batch* para 1024 exemplos e recorreu-se à técnica da taxa de aprendizado cíclica de perfil triangular proposta por Smith (2017) cujo diagrama está exposto na Figura 8. Para os casos simulados, empregou-se a seguinte configuração:  $lr_{\min} = 10^{-5}$ ,  $lr_{\max} = 10^{-3}$  e passo = 64.

Figura 8 – Taxa de aprendizado cíclica de perfil triangular que impede que a evolução da minimização de  $J(\theta)$  fique retida em pontos de sela ou mínimos locais.



Fonte: Elaboração do autor (2020).

Em relação ao conjunto de treinamento, gerou-se aproximadamente  $10^6$  (ou seja,  $10^3$  épocas) exemplos de padrões de erros com probabilidade de transição  $\epsilon = 0.3$  e pesos de Hamming menores ou iguais a  $t + 1$ . Na etapa de teste, os padrões de erro foram simulados com  $\epsilon \in [0.05, 0.3]$  sem qualquer restrição dos pesos de Hamming<sup>7</sup>.

As simulações foram realizadas para os códigos BCH elencados na tabela 1. Para a obtenção dos desempenhos dos decodificadores em questão, avaliou-se as taxas de erro de bloco (*Frame Error Rate*, FER) alcançados para os códigos da Tabela 1 usando o método de Monte Carlo com um limite máximo de 100 erros de bloco detectados para cada valor de  $\epsilon$ .

As Figuras 9 e 10 apresentam as performances obtidas com esses decodificadores, onde DNN-TC indica os resultados usando o decodificador projetado por Tallini e Cull (1995), enquanto DNN-NM refere-se àqueles nos quais o decodificador *Neural-Max* fora empregado. Além disso, nas mesmas figuras, estão plotadas as curvas ML (*Maximum Likelihood*) que representam os máximos desempenhos teóricos e os resultados BDD (*Bounded Distance Decoding*) que, por sua vez, correspondem ao critério de decodificação que decide pela única palavra-código  $c$  que atenda a condição  $d(\mathbf{r}, c) \leq t$ ; caso essa condição não seja atendida, a decodificação falha. Para efeitos de compara-

<sup>7</sup>O uso de uma probabilidade de transição fixa e a restrição dos pesos de Hamming para a geração de exemplos na etapa de treinamento são procedimentos propostos por Tallini e Cull (1995).

ção, também são expostas as curvas que indicam quando nenhum esquema/método de codificação é usado na transmissão da informação.

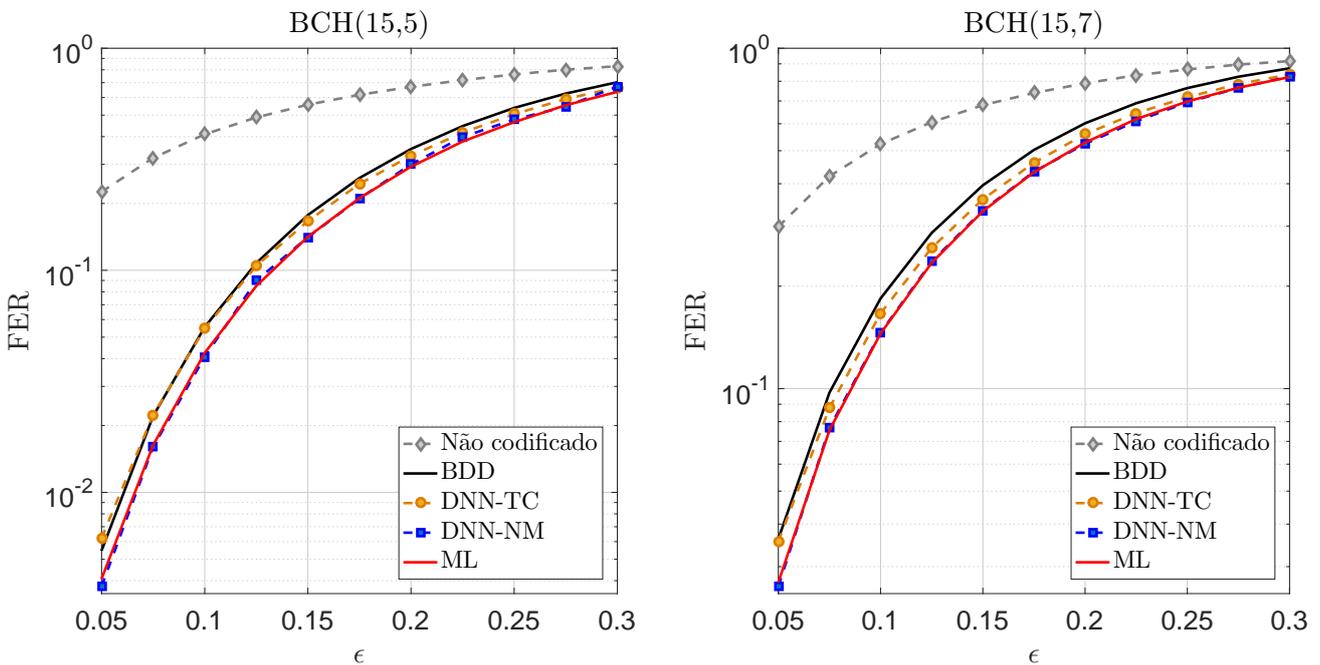
Tabela 1 – Códigos BCH avaliados para uma transmissão usando o BSC.

$n$	$k$	$t$
15	7	2
	5	3
31	16	3
	11	5

Fonte: Elaboração do autor (2020).

De imediato, os resultados obtidos reforçam que, independentemente do comprimento do código, a estratégia de codificação é fundamental para reduzir a probabilidade de erro da informação recebida. Além disso, verifica-se que, para todos os códigos em análise, o desempenho do decodificador proposto por Tallini e Cull (1995) usando a DNN da Figura 7 supera ou é equivalente à performance relativa ao BDD.

Figura 9 – Curvas de desempenhos obtidos com a DNN usando o decodificador proposto por Tallini e Cull (1995) e o decodificador Neural-Max para os códigos BCH(15,5) e BCH(15,7).

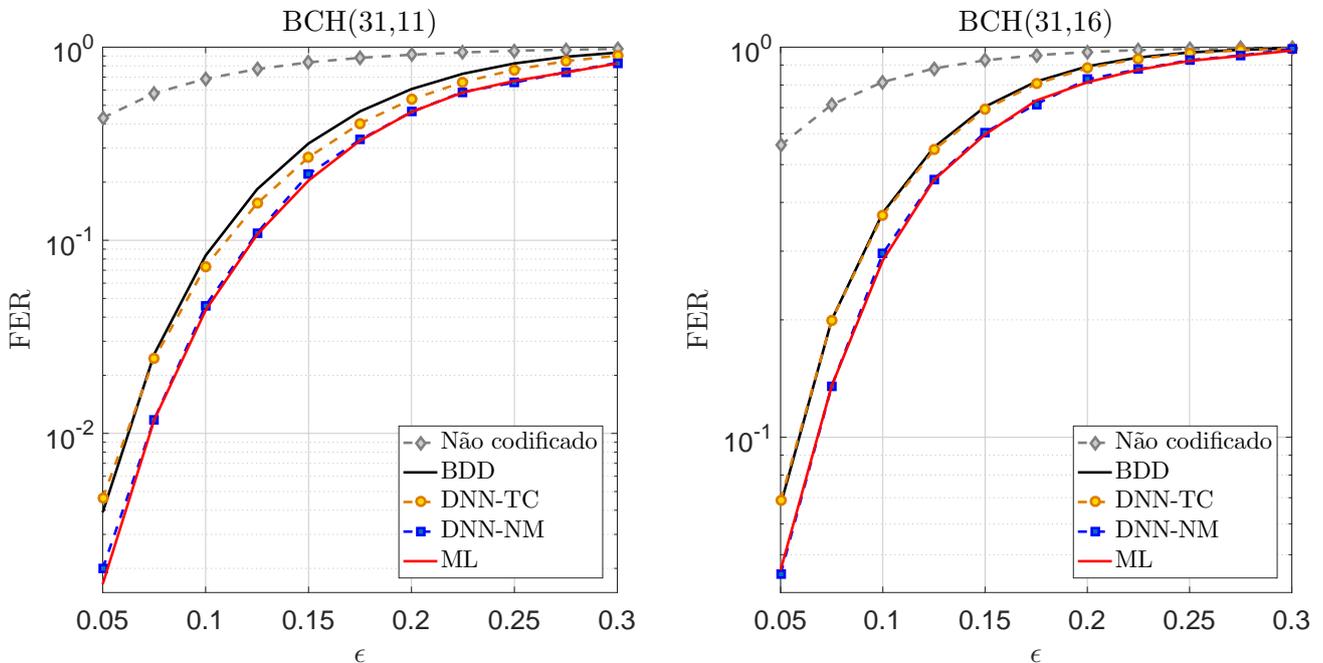


Fonte: Elaboração do autor (2020).

De fato, embora obtenha bons desempenhos, a estratégia usada por Tallini e Cull (1995) não alcança as curvas ML. Visualiza-se ainda que as discrepâncias entre ambos desempenhos (DNN-TC e ML) tendem a crescer de acordo com o comprimento do código. Esse fenômeno pode ser justificado pelo fato de que os vetores síndromes de tamanho  $n - k$  estão diretamente relacionados

aos comprimentos dos códigos; portanto, para maiores valores de  $n$ , maior também é a quantidade de inconsistências dos exemplos de treinamento discutidos na seção 6.

Figura 10 – Curvas de desempenhos obtidos com a DNN usando o decodificador proposto por Tallini e Cull (1995) e o decodificador Neural-Max para os códigos BCH(31,11) e BCH(31,16).



Fonte: Elaboração do autor (2020).

Nota-se que o decodificador NM não tem seu desempenho afetado pelo comprimento do código; além disso, consegue superar — em todos os casos analisados — as performances obtidas com o decodificador de Tallini e Cull (1995), inclusive alcançando o máximo desempenho possível (ML). Desse modo, os resultados alcançados indicam que, com a obtenção iterativa das posições do padrão de erro a partir da predição inicial de uma rede neural profunda e baseado na confiabilidade de cada posição desse vetor, é possível atingir uma decodificação satisfatória.

## 9 Considerações finais

No decorrer desse artigo, abordou-se o uso de redes neurais como uma alternativa para esquemas/projetos de decodificação de códigos de comprimento curto. Conforme exposto, devido aos requisitos específicos de comunicação por parte de tecnologias emergentes, esses códigos têm se manifestado como ótimos candidatos para atender tais demandas. Entretanto, para um desempenho razoável, os tradicionais métodos de decodificação são computacionalmente custosos para lidar com os códigos mais promissores (BCHs).

Nesse contexto, estendeu-se, para códigos BCH com  $n \leq 31$ , o projeto de decodificador proposto na literatura que aplica a síndrome do vetor recebido no processo de treinamento/inferência da rede neural. De fato, atestou-se que usando padrões de erro com  $t + 1$  na fase de treinamento, uma rede densa do tipo *feedforward* é capaz de superar a decodificação BDD; entretanto, observou-se que além de não alcançar o desempenho ML, a performance dessa abordagem tende a piorar para códigos com comprimentos maiores.

Com efeito, o decodificador *Neural-Max* proposto não tem seu desempenho afetado pelo comprimento do código. Ademais, o novo decodificador obteve os melhores resultados, inclusive, aproximando-se do desempenho ML para todos os casos, indicando sua eficiência e robustez para os códigos analisados. Por fim, é válido mencionar que o *Neural-Max* não exige qualquer alteração da etapa de treinamento da rede e possui flexibilidade quanto à rede neural implementada.

## Referências

BISHOP, C. M. **Pattern Recognition and Machine Learning**. Springer, 2006.

BOSE, R. C.; RAY-CHAUDHURI, D. K. Further results on error correcting binary group codes. **Information and Control**, v. 3, n. 3, p. 279-290, set. 1960a. DOI: [http://doi.org/10.1016/S0019-9958\(60\)90870-6](http://doi.org/10.1016/S0019-9958(60)90870-6).

BOSE, R. C.; RAY-CHAUDHURI, D. K. On a class of error correcting binary group codes. **Information and Control**, v. 3, n. 1, p. 68-79, mar. 1960b. DOI: [http://doi.org/10.1016/S0019-9958\(60\)90287-4](http://doi.org/10.1016/S0019-9958(60)90287-4).

DURISI, G.; KOCH, T.; POPOVSKI, P. Toward massive, ultrareliable, and low-latency wireless communication with short packets. **Proceedings of the IEEE**, v. 104, n. 9, p. 1711-1726, set. 2016. DOI: <http://doi.org/10.1109/JPROC.2016.2537298>.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. Disponível em: <http://www.deeplearningbook.org>. Acesso em: 14 maio 2020.

GRUBER, T.; CAMMERER, S.; HOYDIS, J.; BRINK, S. On deep learning-based channel decoding. In: ANNUAL CONFERENCE ON INFORMATION SCIENCES AND SYSTEMS, 2017, Baltimore. **Proceedings ...**, s.L.: IEEE, 2017. p. 1-6. DOI: <http://doi.org/10.1109/CISS.2017.7926071>.

HAYKIN, S. **Digital Communication Systems**. Hoboken, NJ: John Wiley & Son, Inc., 2014.

HOCQUENGHEM, A. Codes correcteurs d'erreurs. **Chiffres**, v. 2, p. 147-156, 1959. Disponível em: <http://kom.aau.dk/~heb/kurser/NOTER/KOFA02.PDF>. Acesso em: 22 mar. 2020.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. **Neural Networks**, v. 2, n. 5, p. 359-366, 1989. DOI: [https://doi.org/10.1016/0893-6080\(89\)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8).

IOFFE, S.; SZEGEDY, C. Batch normalization: accelerating deep network training by reducing internal covariate shift. **CoRR**, abs/1502.03167, 2015. Disponível em: <http://arxiv.org/abs/1502.03167>. Acesso em: 28 abr. 2020.

KAMASSURY, J. K. S.; SILVA, V. F. O. Rápido reconhecimento de modulações analógicas e digitais via redes residuais profundas. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA FÍSICA, 14, Ponta Grossa, 2020. **Anais...** Ponta Grossa: Atena Editora, 2020. cap. 9, p. 83-97. DOI: <http://doi.org/10.22533/at.ed.1572002039>.

KAMASSURY, J. K. S.; TÔRRES, I. F.; DUARTE, W. G. Decodificação de máxima verossimilhança para códigos de bloco lineares: probabilidades de erro do código de repetição e do código de Hamming. **REMAT: Revista Eletrônica da Matemática**, v. 5, n. 2, p. 177-191, 1 jul. 2019. DOI: <https://doi.org/10.35819/remat2019v5i2id3371>.

KINGMA, D. P.; BA, J. Adam: a method for stochastic optimization. **CoRR**, abs/1412.6980, 2014. Disponível em: <http://arxiv.org/abs/1412.6980>. Acesso em: 28 abr. 2020.

LIN, S.; COSTELLO, D. **Error control coding: Fundamentals and Applications**. 2. ed. Upper Saddle River, NJ: Prentice-Hall, 2004.

O'SHEA, T.; HOYDIS, J. An introduction to deep learning for the physical layer. **IEEE Transactions on Cognitive Communications and Networking**, v. 3, n. 4, p. 563-575, dez. 2017. DOI: <https://doi.org/10.1109/TCCN.2017.2758370>.

PROAKIS, J. G.; SALEHI, M. **Digital Communication**. 5. ed. New York: McGraw-Hill, 2007.

SHIRVANIMOGHADDAM, M.; MOHAMMADI, M. S.; ABBAS, R.; MINJA, A.; YUE, C.; MATUZ, B.; HAN, G.; LIN, Z.; LIN, W.; LI, Y.; JONHSON, S.; VUCETIC, B. Short block-length codes for

ultra-reliable low latency communications. **IEEE Communications Magazine**, v. 57, n. 2, p. 130-137, fev. 2019. DOI: <http://doi.org/10.1109/MCOM.2018.1800181>.

SMITH, L. N. Cyclical learning rates for training neural networks. In: IEEE WINTER CONFERENCE ON APPLICATIONS OF COMPUTER VISION, 2017, Santa Rosa. **Proceedings...**, s.L.: IEEE, 2017. p. 464-472. DOI: <http://doi.org/10.1109/WACV.2017.58>.

TALLINI, L. G.; CULL, P. Neural nets for decoding error-correcting codes. In: IEEE TECHNICAL APPLICATIONS CONFERENCE AND WORKSHOPS. NORTHCON/95. CONFERENCE RECORD, 1995, Portland. **Proceedings...**, s.L.: IEEE, 2002. p. 89-94. DOI: <http://doi.org/10.1109/NORTHCON.1995.485019>.

WANG, X.-A; WICKER, S. B. An artificial neural net Viterbi decoder. **IEEE Transactions on Communications**, v. 44, n. 2, p. 165-171, 1996. DOI: <http://doi.org/10.1109/26.486609>.

WHITE, H. Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings. **Neural Networks**, v. 3, n. 5, p. 535-549, 1990. DOI: [https://doi.org/10.1016/0893-6080\(90\)90004-5](https://doi.org/10.1016/0893-6080(90)90004-5).