UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de São José do Rio Preto

André Furlan

Autenticação Biométrica de Locutores Drasticamente Disfônicos Aprimorada pela *Imagined Speech*

São José do Rio Preto
2022

André Furlan

Autenticação Biométrica de Locutores Drasticamente Disfônicos Aprimorada pela *Imagined Speech*

Tese apresentada como parte dos requisitos para obtenção do título de Doutor em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista 'Júlio de Mesquita Filho", Campus de São José do Rio Preto.

Orientador: Prof. Dr. Rodrigo Capobianco Guido

São José do Rio Preto

2022

ficha catalografica

André Furlan

Autenticação Biométrica de Locutores Drasticamente Disfônicos Aprimorada pela *Imagined Speech*

Tese apresentada como parte dos requisitos para obtenção do título de Doutor em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista 'Júlio de Mesquita Filho", Campus de São José do Rio Preto.

Comissão Examinadora

Prof. Dr. Rodrigo Capobianco Guido
UNESP – Câmpus de São José do Rio Preto
Orientador

Prof. Dr. Exemplo Jr
Universidade – Câmpus

Prof. Dr. Exempl2
Universidade – Câmpus

São José do Rio Preto
06 de Agosto de 2022

Haha…

Agradecimentos

Agradece

"Citação"

-O Autor.

Resumo

Resumo

Abstract

Summary

Abreviações

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

1.1 Considerações Iniciais e Objetivos

A autenticação baseada em biometria vem despertando um interesse crescente, uma vez que se aproveita de características biométricas altamente correlacionadas com o indivíduo, resultando em um nível mais elevado de segurança. Algumas biometrias mostraram-se discriminatórias em relação a certos grupos, como os indivíduos com deficiências físicas, que podem enfrentar dificuldades ao utilizá-las. Esse é o caso de boa parte dos tipos de biometrias utilizadas, incluindo a fala que é um dos focos desse estudo. Por outro lado, o eletroencefalograma (EEG) evita tais problemas, ao mesmo tempo em que acrescenta algumas características únicas, como a exploração das características neurais do usuário, criando, assim, uma contramedida contra pressões externas e complementando eventuais falhas ou faltas de outros métodos.

1.1.1 Objetivos

Comparar o desempenho das estratégias de *feature learnning* baseadas em *auto-encoders* com técnicas de análise como as *Wavelet-Packet* de Tempo Discreto usando a engenharia paraconsistente de características.

O problema-alvo deste projeto é conceber algoritmos biométricos para autenticar, em princípio por meio da fala, indivíduos com locuções severamente degradadas complementando tais informações com aquelas provenientes dos sinais cerebrais extraídos durante a fonação (imagined speech).

Estudar base de dados existentes, criar a própria base de dados. Depois iniciar com a criação de vetores de características usando *autoencoders* e análise com *wavelets packet tranform*, e classificá-los com redes neurais profundas residuais e recorrentes.

Text-dependent: Uma mesma frase é falada e imaginada

Text-independent: Pode ser falada qualquer coisa tanto no treinamento quanto nos testes.

1.2 Estrutura do trabalho

2 Revisão de Bibliográfica

2.1 Conceitos utilizados

2.1.1 Sinais digitais e sub-amostragem (*downsampling*)

Os sinais digitais, tanto de voz quanto aqueles vindos das medições de Eletroencefalograma (ECG), isto é, aqueles que estão amostrados e quantizados (HAYKIN; MOHER, 2011), constituem a base deste trabalho. Além do processo de digitalização, inerente ao ato de armazenar sinais em computadores, os mesmos podem sofrer, a depender da necessidade ou possibilidade, sub-amostragens ou *downsamplings* (POLIKAR *et al.*, 1996). Isso implica em uma estratégia de redução de dimensão e, comumente, ocorre após a conversão de domínio dos sinais com base em filtros digitais do tipo *wavelet*, a serem apresentados adiante. Um exemplo consta na Figura 1, na qual as partes pretas contêm dados e as brancas representam os elementos removidos. Tendo em vista que este trabalho está baseado em sinais digitais de voz e ECG com base em *wavelets*, o processo de sub-amostragem é essencial.

Figura 1 – Sub-amostragem

Fonte: O autor

2.1.2 Caracterização dos processos de produção da voz humana

A fala possui três grandes áreas de estudo: A fisiológica, também conhecida como fonética articulatória, a acústica, referida como fonética acústica, e ainda, a perceptual, que cuida da percepção da fala (KREMER; GOMES, 2014). Neste trabalho, o foco será apenas na questão acústica, pois não serão analisados aspectos da fisiologia relacionada à voz, mas sim os sinais sonoros propriamente ditos.

2.1.2.1 Sinais vozeados *versus* não-vozeados

Quando da análise dos sinais de voz, consideram-se as partes vozeadas e não-vozeadas. Aquelas são produzidas com a ajuda da vibração quase periódica das pregas vocais, enquanto estas praticamente não contam com participação regrada da referida estrutura.

2.1.2.2 Frequência fundamental da voz

Também conhecida como F_0, é o componente periódico resultante da vibração das pregas vocais. Em termos de percepção, se pode interpretar F_0 como o tom da voz, isto é, a frequência de *pitch* (KREMER; GOMES, 2014). Vozes agudas tem uma frequência de *pitch* alto, enquanto vozes mais graves tem baixa. A alteração da frequência (jitter) e/ou intensidade (shimmer) do *pitch* durante a fala é definida como entonação, porém, também pode indicar algum distúrbio ou doença relacionada ao trato vocal (WERTZNER; SCHREIBER; AMARO, 2005).

A frequência fundamental da voz é o número de vezes na qual uma forma de onda característica, que reflete a excitação pulmonar moldada pelas pregas vocais, se repete por unidade de tempo. Sendo assim, as medidas de F_0 geralmente são apresentadas em Hz (FREITAS, 2013).

A medição de F_0 está sujeita a contaminações surgidas das variações naturais de *pitch* típicas da voz humana (FREITAS, 2013). A importância de se medir F_0 corretamente vem do fato de que, além de carregar boa parte da informação da fala, ela é a base para construção das outras frequências que compõe os sinais de voz, que são múltiplas de F_0.

2.1.2.3 Formantes

O sinal de excitação que atravessa as pregas vocais é rico em harmônicas, isto é, frequências múltiplas da fundamental. Tais harmônicas podem ser atenuadas ou amplificadas, em função da estrutura dos tratos vocal e nasal de cada locutor. Particularmente, o primeiro formante (F_1), relaciona-se à amplificação sonora na cavidade oral posterior e à posição da língua no plano vertical; o segundo formante (F_2) à cavidade oral anterior e à posição da língua no plano horizontal; o terceiro formante (F_3) relaciona-se às cavidades à frente e atrás do ápice da língua e, finalmente, o quarto formante (F_4) relaciona-se ao formato da laringe e da faringe na mesma altura (VALENÇA *et al.*, 2014). Formantes caracterizam fortemente os locutores, pois cada indivíduo possui um formato de trato vocal e nasal. Assim, tais frequências, que podem ser capturadas com ferramentas diversas, a exemplo da Transformada *Wavelet*, são de suma importância na área de verificação de locutores.

2.1.3 Escalas e energias dos sinais

A energia de um sinal digital $s[\cdot]$ com M amostras é definida como

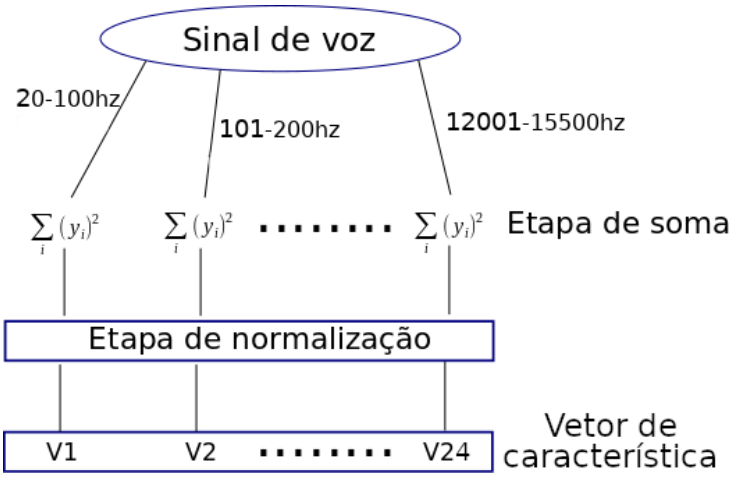
$$E = \sum_{i=0}^{M-1} (s_i)^2 \qquad . \tag{2.1}$$

E pode ainda sofrer normalizações e ter a sua mensuração restrita a uma parte específica do sinal sob análise. Possibilidades para tais restrições podem, por exemplo, envolver a escala BARK (ZWICKER, 1961) e MEL (BERANEK, 1949) que serão utilizadas neste trabalho.

2.1.3.1 A escala BARK

BARK foi definida tendo em mente vários tipos de sinais acústicos. Essa escala corresponde ao conjunto de 25 bandas críticas da audição humana. Suas frequências-base de audiometria são, em Hz: **20, 100, 200, 300, 400, 510, 630, 770, 920, 1080, 1270, 1480, 1720, 2000, 2320, 2700, 3150, 3700, 4400, 5300, 6400, 7700, 9500, 12000, 15500**. Nessa escala,os sinais digitais no domínio temporal atravessam filtros passa-faixas (BOSI; GOLDBERG, 2002) para os quais o início e o final da banda de passagem correspondem à frequências-base consecutivas resultando em um vetor de características com 24 coeficientes e, em seguida, as energias dos sinais filtrados são utilizadas como características descritivas de propriedades do sinal sob análise, como mostrado na Figura 2.

Figura 2 – Cálculo de vetores de características com BARK



Fonte: O autor

2.1.3.2 A escala MEL

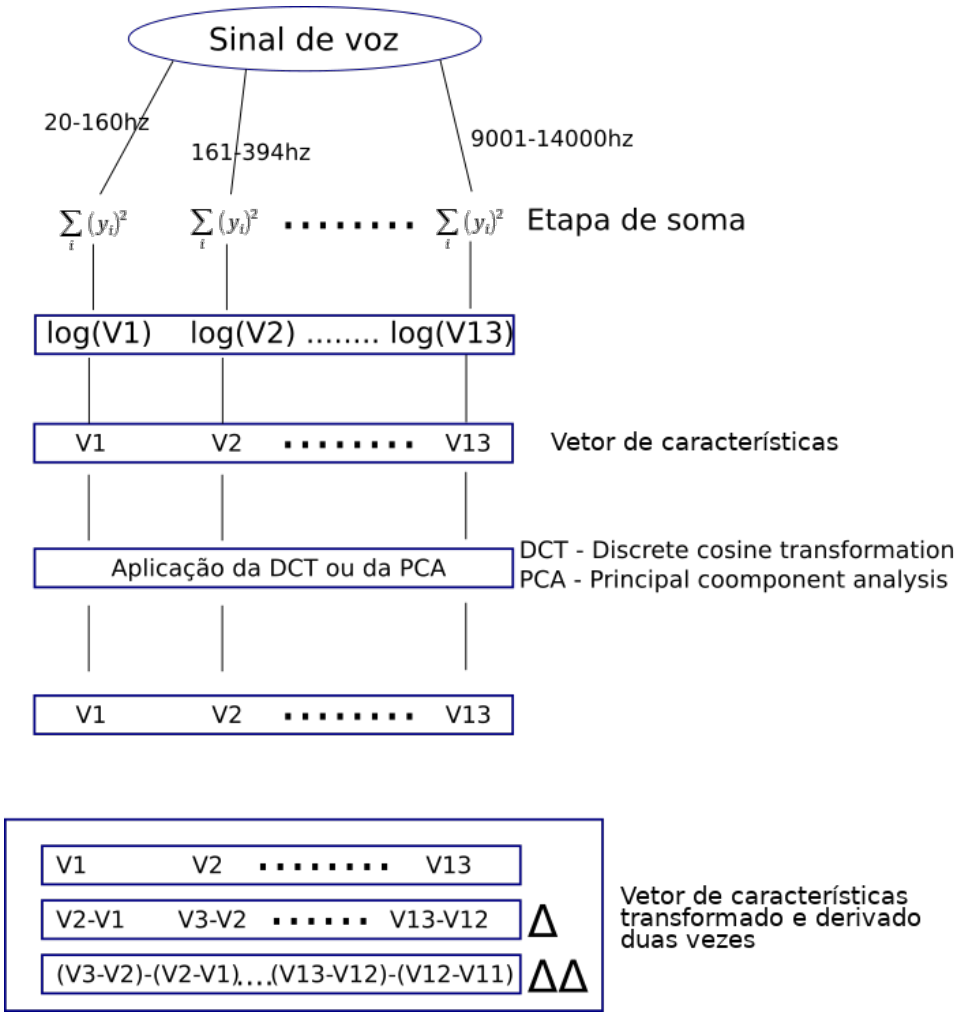
Escala Mel, advinda do termo *melody*, é uma adaptação da escala Bark para sinais de voz. Dentre as várias implementações de bandas críticas a escolhida foi a implementação que contém os valores em Hz: **20, 160, 394, 670, 1000, 1420, 1900, 2450, 3120, 4000, 5100, 6600, 9000, 14000**.

A variante que será usada neste trabalho é conhecida como *Mel-frequency cepstral coefficients*(MFCC) a qual inclui, além dos intervalos definidos, uma diminuição da correlação entre os componentes gerados via aplicação da Transformada Discreta Cosseno

(DCT) (SALOMON; MOTTA; BRYANT, 2007) ou da Análise de Componentes Principais (PCA) (JOLLIFFE, 2002) seguida de duas derivações no vetor de características resultando em um total de 11 coeficientes. Nesse trabalho foi escolhida a DCT, no entanto, PCA poderia também ser escolhida sem prejuízos, o uso de uma ou outra depende da preferência do autor.

Novamente, desconsiderando qualquer etapa intermediária que possa ser adicionada, as energias calculadas nos intervalos definidos na escala MEL podem, por si mesmas, constituir um vetor de características, como mostrado na Figura 2.

Figura 3 – Cálculo de vetores de características com MEL



Fonte: O autor

2.1.4 Filtros digitais *wavelet*

Filtros digitais *wavelet* têm sido utilizados com sucesso para suprir as deficiências de janelamento de sinal apresentadas pelas Transformadas de Fourier e de Fourier de Tempo Reduzido. *Wavelets* contam com variadas funções-filtro e têm tamanho de janela variável, o que permite uma análise multirresolução (ADDISON; WALKER; GUIDO,

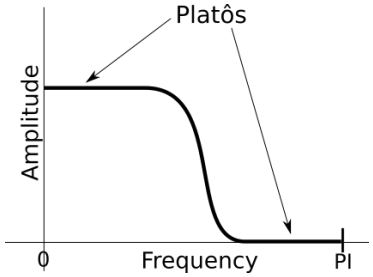
2009). Particularmente, as *wavelets* proporcionam a análise do sinal de forma detalhada tanto no espectro de baixa frequência quanto no de alta contando com diferentes funções-base não periódicas diferentemente da tradicional transformada de Fourrier que utilizam somente as bases periódicas senoidal e cossenoidal.

É importante observar que, quando se trata de Transformadas *Wavelet*, seis elementos estão presentes: dois filtros de análise, dois filtros de síntese e as funções ortogonais *scaling* e *wavelet*. No tocante a sua aplicação, só a transformada direta, e não a inversa, será usada na construção dos vetores de características. Portanto, os filtros de síntese, a função *scaling* e a função *wavelet* não serão elementos abordados aqui: eles somente interessariam caso houvesse a necessidade da transformada inversa.

No contexto dos filtros digitais baseados em *wavelets*, o tamanho da janela recebe o nome de **suporte**. Janelas definem o tamanho do filtro que será aplicado ao sinal. Quando esse é pequeno (limitado), se diz que a janela tem **um suporte compacto** (POLIKAR *et al.*, 1996).

Se diz que uma *wavelet* tem boa **resposta em frequência** quando, na aplicação da mesma para filtragem, não são causadas muitas pertubações indesejadas ao sinal, no domínio da frequência. Os filtros *wavelet* de Daubechies (DAUBECHIES, 1992) se destacam nesse quesito por serem *maximamente planos* (*maximally-flat*) (BUTTERWORTH, 1930) (BIANCHI, 2007) nos platôs de resposta em frequência como indicado na Figura 4 ao contrário do que ocorre na Figura 5.

Figura 4 – Platôs maximamente planos em um filtro digital: característica da família de Daubechies

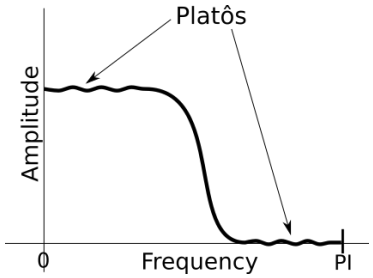


Fonte: O autor

Além da resposta em frequência, na aplicação de um filtro digital *wavelet* também é possível considerar a **resposta em fase**, que constitui um atraso ou adiantamento do sinal filtrado em relação ao sinal original, ambos no domínio temporal. Esse deslocamento pode ser **linear**, **quase linear** ou **não linear**:

- na resposta em fase **linear**, há o mesmo deslocamento de fase para todos os componentes do sinal;

Figura 5 – Platôs não maximamente planos de um filtro digital: características de outros filtros *wavelet*, distintos da família de Daubechies



Fonte: O autor

- quando a resposta em fase é **quase linear** existe uma pequena diferença no deslocamento dos diferentes componentes do sinal;
- finalmente, quando a resposta é **não linear**, acontece um deslocamento significativamente heterogêneo para as diferentes frequências que compõe o sinal.

Idealmente, é desejável que todo filtro apresente boa resposta em frequência e em fase linear. Características de fase e frequência de algumas famílias de filtros *wavelet* constam na Tabela Tabela 1.

Tabela 1 – Algumas das *wavelets* mais usadas e suas propriedades

Wavelet	**Resposta em frequência**	**Resposta em fase**
Haar	Pobre	Linear
Daubechies	mais próxima da ideal à medida que o suporte aumenta; *maximally-flat*	Não linear
Symmlets	mais próxima da ideal à medida que o suporte aumenta; não *maximally-flat*	Quase linear
Coiflets	mais próxima da ideal à medida que o suporte aumenta; não *maximally-flat*	Quase linear

Fonte: Elaborado pelo autor, 2023.

2.1.4.1 O algoritmo de Mallat para a Transformada *Wavelet*

Baseando-se no artigo (GUIDO, 2015), percebe-se que algoritmo de Mallat faz com que aplicação das *wavelets* seja uma simples multiplicação de matrizes. O sinal que deve ser transformado se torna uma matriz linear vertical. Os filtros passa-baixa e passa-alta tornam-se, nessa ordem, linhas de uma matriz quadrada que será completada segundo regras que serão mostradas mais adiante. É importante que essa matriz quadrada tenha a mesma dimensão que o sinal a ser transformado.

Interessantemente, para que seja possível a transformação *wavelet*, basta ter disponível o vetor do filtro passa-baixas calculado a partir da *mother wavelet*, que é a função

geradora desse filtro, já que o passa-alta pode ser construído a partindo-se da ortogonalidade do primeiro.

Determinar a ortogonal de um vetor significa construir um vetor, tal que, o produto escalar do vetor original com sua respectiva ortogonal seja nulo.

Considerando $h[\cdot]$ como sendo o vetor do filtro passa-baixas e $g[\cdot]$ seu correspondente ortogonal, tem-se que $h[\cdot] \cdot g[\cdot] = 0$.

Portanto, se $h[\cdot] = [a, b, c, d]$ então seu ortogonal será $g[\cdot] = [d, -c, b, -a]$ pois:

$$h[\cdot]\cdot g[\cdot] = [a, b, c, d]\cdot[d, -c, b, -a] = (a\cdot d)+(b\cdot(-c))+(c\cdot b)+(d\cdot(-a)) = ad-ad+bc-bc = 0 \quad .$$

A título de exemplo, considera-se:

- o filtro passa baixa baseado na *wavelet* Haar: $h[\cdot] = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]$
- o seu respectivo vetor ortogonal: $g[\cdot] = [\frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}]$
- e também o seguinte sinal-exemplo de entrada: $s = \{1, 2, 3, 4\}$

Se o tamanho do sinal a ser tratado é quatro e se pretende-se aplicar o filtro Haar, a seguinte matriz de coeficientes é construída:

$$\begin{pmatrix} \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0 \\ \frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}, 0, 0 \\ 0, 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \\ 0, 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \end{pmatrix} \tag{2.2}$$

Tendo em vista que a dimensão do sinal sob análise é diferente da dimensão do filtro, basta completar cada uma das linhas da matriz de coeficientes com zeros. A matriz é montada de forma que ela seja ortogonal.

Montada a matriz de filtros, segue-se com os cálculos da transformada:

$$\begin{pmatrix} \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0 \\ \frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}, 0, 0 \\ 0, 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \\ 0, 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} = \begin{pmatrix} \frac{3}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \\ \frac{7}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix} \tag{2.3}$$

Realizada a multiplicação, é necessário montar o sinal filtrado. Isso é feito escolhendo, dentro do resultado, valores alternadamente de forma que o vetor resultante seja:

$$resultado = \Big[\frac{3}{\sqrt{2}}, \frac{7}{\sqrt{2}}, \frac{-1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}\Big] \quad . \tag{2.4}$$

Percebe-se que, na transformação descrita nas Equações Equação 2.2, Equação 2.3 e Equação 2.4, a **aplicação dos filtros sobre o vetor de entrada ocorreu apenas uma vez**. Sendo assim, se diz que o sinal recebeu uma **transformação de nível 1**. A cada transformação, há uma separação do sinal em dois componentes: o de baixa e o de alta frequência.

Embora haja um limite, que será mencionado adiante, é possível aplicar mais de um nível de decomposição ao sinal. Para que se possa fazer isso, a Transformada *Wavelet* nível 2 deve considerar apenas a parte de baixas frequências da primeira transformada; a transformada de nível 3 deve considerar apenas a parte de baixas frequências da transformada nível 2, e assim consecutivamente.

Nos exemplos numéricos mostrados nas Tabelas Tabela 2, Tabela 3 e Tabela 4, usou-se um filtro normalizado cujos coeficientes são $\{\frac{1}{2}, -\frac{1}{2}\}$. Os dados destacados em **verde** correspondem ao **vetor original** que será tratado. Cada uma das linhas são os resultados das transformações nos níveis 1, 2, 3 e 4, respectivamente. As partes em **azul** correspondem à porção de **baixas frequências**, enquanto que as partes em **amarelo** correspondem às porções de **altas frequências**.

Percebe-se que na Tabela Tabela 2, a partir da transformação nível 2, apenas as partes de baixa frequência são modificadas. Isso implica que, no momento da implementação do algoritmo de Mallat **para níveis maiores que 1**, a abordagem será **recursiva**. Em outras palavras, a partir do nível 1 se deve aplicar Mallat apenas às porções de baixas-frequências geradas pela transformação anterior.

Tabela 2 – Exemplo numérico da transformação *wavelet* aplicada a um vetor

Sinal	**32**	**10**	**20**	**38**	**37**	**28**	**38**	**34**	**18**	**24**	**24**	**9**	**23**	**24**	**28**	**34**
Nível 01	21	29	32,5	36	21	16,5	23,5	31	11	-9	4,5	2	-3	7,5	-0,5	-3
Nível 02	25	34,25	18,75	27,25	-4	-1,75	2,25	-3,75	11	-9	4,5	2	-3	7,5	-0,5	-3
Nível 03	29,62	23	-4,625	-4,25	-4	-1,75	2,25	-3,75	11	-9	4,5	2	-3	7,5	-0,5	-3
Nível 04	26,3125	3,3125	-4,625	-4,25	-4	-1,75	2,25	-3,75	11	-9	4,5	2	-3	7,5	-0,5	-3

Fonte: Elaborado pelo autor, 2023.

2.1.4.2 O algoritmo de Mallat e a Transformada *Wavelet-Packet*

Na Transformada *Wavelet-Packet*, os filtros aplicados são os mesmos da Transformada *Wavelet* e o procedimento recursivo de cálculo também é o mesmo, no entanto, realizada a transformação de nível 1, a transformada de nível 2 deve ser aplicada aos componentes de baixa e de alta frequência. Sendo assim a Transformada *Wavelet-Packet* obtém um nível de detalhes em todo o espectro de frequência, maior do que uma transformação regular.

Os exemplos mostrados nas Tabelas Tabela 3 e Tabela 4 permitem perceber como se dão as transformações na porção de **baixa** e de **alta** frequências, respectivamente, após a transformação *wavelet-packet* de nível 1, 2, 3 e 4.

Devido ao *downsampling* aplicado às porções de alta frequência, essas partes acabam por ficar "espelhadas" no espectro (JENSEN; COUR-HARBO, 2001), ou seja, suas sequências ficam invertidas. Para resolver esse problema e preservar a ordem das subbandas no sinal transformado, os filtros são aplicados em ordem inversa nas porções de alta frequência. Isso altera como o algoritmo de Mallat deve ser implementado para a Transformada *Wavelet-Packet*, já que dessa vez é preciso se atentar a ordem da aplicação dos filtros passa-alta e passa-baixa.

Tabela 3 – Exemplo numérico de *wavelet-packet* Haar aplicada ao vetor da Tabela 2 (porção das baixas frequências)

Sinal	**32**	**10**	**20**	**38**	**37**	**28**	**38**	**34**
Nivel 01	21	29	32,5	36	21	16,5	23,5	31
Nivel 02	25	34,25	18,75	27,25	-4	-1,75	2,25	-3,75
Nivel 03	29,62	23	-4,625	-4,25	-1,125	3	-2,875	-0,75
Nivel 04	26,3125	3,3125	-0,1875	-4,4375	0,9375	-2,0625	-1,0625	-1,8125

Fonte: Elaborado pelo autor, 2023.

Tabela 4 – Exemplo numérico de *wavelet-packet* Haar aplicada ao vetor da Tabela 2 (porção das altas frequências)

Sinal	**18**	**24**	**24**	**9**	**23**	**24**	**28**	**34**
Nivel 01	11	-9	4,5	2	-3	7,5	-0,5	-3
Nivel 02	10	1,25	-5,25	1,25	1	3,25	2,25	-1,75
Nivel 03	5,625	-2	4,375	-3,25	-1,125	2	2,125	0,25
Nivel 04	1,8125	3,8125	3,8125	0,5625	0,4375	-1,5625	0,9375	1,1875

Fonte: Elaborado pelo autor, 2023.

2.1.5 Engenharia Paraconsistente de características

Nos processos de classificação, frequentemente surge a questão: "Os vetores de características criados proporcionam uma boa separação de classes?". A Engenharia Paraconsistente de Características, recém publicada (GUIDO, 2019), que usa a paraconsistência (COSTA; BÉZIAU; BUENO, 1998), (COSTA; ABE, 2000) é, em meio a outras técnicas, uma ferramenta que pode ser usada para responder essa questão.

O processo inicia-se após a aquisição dos vetores de características para cada classe C_n. Se o número de classes presentes for, por exemplo, quatro então estas poderão ser representadas por C_1, C_2, C_3, C_4.

Em seguida é necessário o cálculo de duas grandezas:

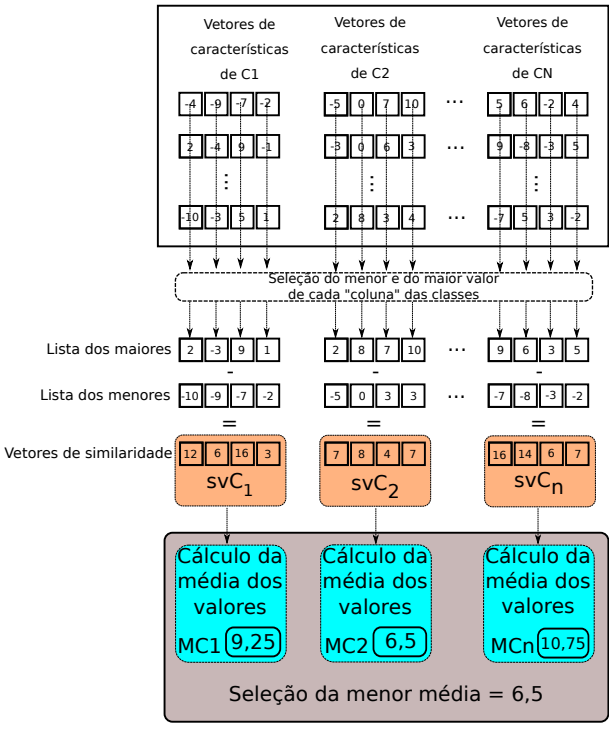
- a menor similaridade intraclasse, α.
- a razão de sobreposição interclasse, β.

α indica o quanto de similaridade os dados têm entre si, dentro de uma mesma classe, enquanto β é a razão de sobreposição entre diferentes classes. Idealmente, α deve ser maximizada e β minimizada para que classificadores extremamente modestos apresentem uma acurácia interessante.

Particularmente, para calcular α e β, é necessária a normalização dos vetores de características de forma que todos os seus componentes estejam no intervalo entre 0 e 1. Em seguida, a obtenção de α se dá selecionando-se os maiores e os menores valores de cada uma das posições de todos os vetores de características para cada classe, gerando assim um vetor para os valores maiores e outro para os menores.

O **vetor de similaridade da classe**(svC_n) é obtido fazendo-se a diferença item-a-item dos maiores em relação aos menores. Finalmente, e para cada classe, é obtida a média dos valores para cada vetor de similaridade, sendo que α é o menor valor dentre essas médias. A Figura 6 contém uma ilustração do processo.

Figura 6 – Cálculo do coeficiente α.



Adaptado de (GUIDO, 2019)

A obtenção de β, assim como ilustrado na Figura 7, também se dá selecionando os maiores e os menores valores de cada uma das posições de todos os vetores de características de cada classe, gerando assim um vetor para os valores maiores e outro para os menores.

Na sequência, realiza-se o cálculo de R cujo valor é a quantidade de vezes que um valor do vetor de características de uma classe se encontra entre os valores maiores e menores de outra classe.

Seja:

- N a quantidades de classes;
- X a quantidade de vetores de características por classe;
- T o tamanho do vetor de características.

Então, F, que é o número máximo de sobreposições possíveis entre classes, é dado por:

$$F = N.(N-1).X.T \quad . \tag{2.5}$$

Finalmente, β é calculado da seguinte forma:

$$\beta = \frac{R}{F} \quad . \tag{2.6}$$

Neste ponto, é importante notar que $\alpha = 1$ sugere fortemente que os vetores de características de cada classe são similares e representam suas respectivas classes precisamente. Complementarmente, $\beta = 0$ sugere os vetores de características de classes diferentes não se sobrepõe (GUIDO, 2019).

Considerando-se o plano paraconsistente (GUIDO, 2019), temos:

- Verdade $\rightarrow$ fé total ($\alpha = 1$) e nenhum descrédito ($\beta = 0$)
- Ambiguidade $\rightarrow$ fé total ($\alpha = 1$) e descrédito total ($\beta = 1$)
- Falsidade $\rightarrow$ fé nula ($\alpha = 0$) e descrédito total ($\beta = 1$)
- Indefinição $\rightarrow$ fé nula ($\alpha = 0$) e nenhum descrédito ($\beta = 0$) .

No entanto, raramente α e β terão valores inteiros como os mostrados na listagem acima: Na maioria das ocasiões, $0 \leqslant \alpha \leqslant 1$ e $0 \leqslant \beta \leqslant 1$. Por isso, se torna necessário o cálculo do **grau de certeza**, isto é, G_1, e do **grau de contradição**, isto é, G_2, conforme segue:

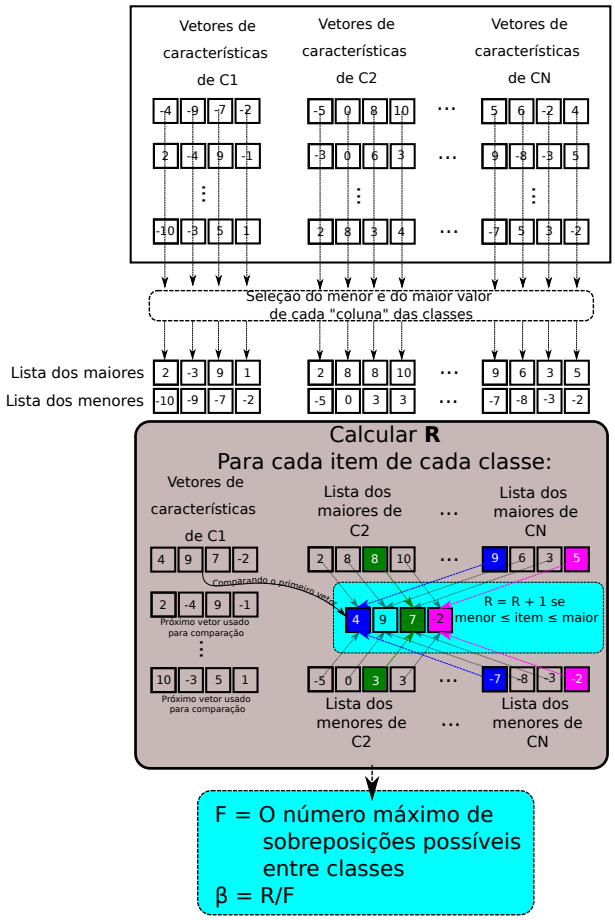
$$G_1 = \alpha - \beta \quad , \tag{2.7}$$

$$G_2 = \alpha + \beta - 1 \quad , \tag{2.8}$$

onde: $-1 \leqslant G_1$ e $1 \geqslant G_2$.

Os valores de G_1 e G_2, em conjunto, definem os graus entre verdade ($G_1 = 1$) e falsidade ($G_1 = -1$) e também os graus entre indefinição ($G_2 = -1$) e ambiguidade

Figura 7 – Cálculo de β: Os itens destacados em azul e rosa são aqueles pertencentes a classe C1 e CN que se sobrepõe, em verde, a sobreposição é entre C1 e C2. Para cada sobreposição verificada soma-se 1 ao valor R. Essa comparação é feita para todos os vetores de características de cada uma das classes.



Adaptado de (GUIDO, 2019)

($G_2 = 1$). Novamente, raramente tais valores inteiros serão alcançados já que G_1 e G_2 dependem de α e β.

O Plano Paraconsistente, para fins de visualização e maior rapidez na avaliação dos resultados, encontra-se ilustrado na Figura 8 e tem quatro arestas precisamente definidas:

- (-1,0) → falsidade;
- (1,0) → verdade;
- (0,-1) → indefinição;
- (0,1) → ambiguidade.

A propósito de ilustração na Figura 8, é possível ver um pequeno círculo indicando os graus dos quatro casos listados.

Para se ter ideia em que área exatamente se encontram as classes avaliadas, as distâncias (D) do ponto $P = (G_1, G_2)$ até o limites supracitados podem ser computadas. Tais cálculos podem ser feitos da seguinte forma:

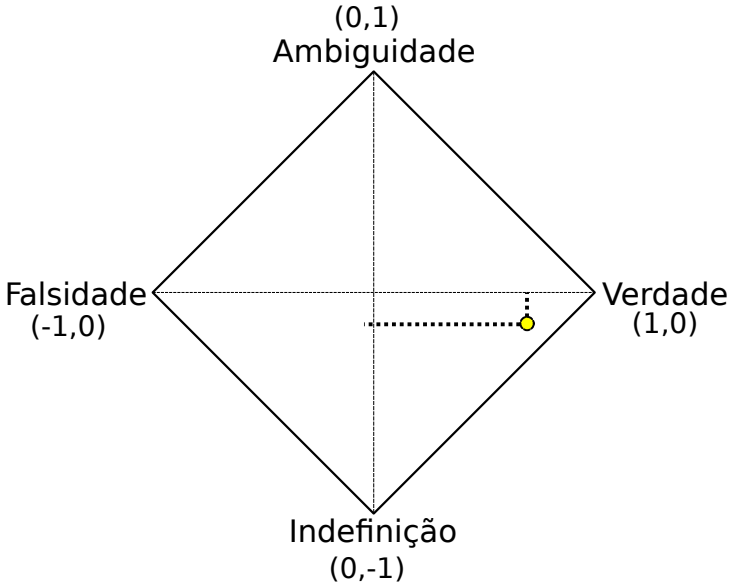
$$D_{-1,0} = \sqrt{(G_1 + 1)^2 + (G_2)^2} \quad , \tag{2.9}$$

$$D_{1,0} = \sqrt{(G_1 - 1)^2 + (G_2)^2} \quad , \tag{2.10}$$

$$D_{0,-1} = \sqrt{(G_1)^2 + (G_2 + 1)^2} \quad , \tag{2.11}$$

$$D_{0,1} = \sqrt{(G_1)^2 + (G_2 - 1)^2} \quad . \tag{2.12}$$

Figura 8 – O plano paraconsistente: O pequeno círculo indica os graus de falsidade(-1,0), verdade(1,0), indefinição(0,-1) e ambiguidade(0,1)



Adaptado de (GUIDO, 2019)

Na prática, ou seja, para fins de classificação, geralmente considera-se a distância em relação ao ponto *"(1,0) → Verdade"*, que é o ponto ótimo: quanto mais próximo o ponto (G_1, G_2) estiver de $(1, 0)$, mais as os vetores de características das diferentes classes estão naturalmente separados. Isso implica, dentro da limitação de cada algoritmo, em resultados melhores sejam quais forem os classificadores usados.

2.1.6 Interfaces Humano-Máquina e EEG

Entre os métodos de Interface Cérebro-Computador (BCI), o Eletroencefalograma (EEG) se destaca como o sistema mais econômico e simples de implementar. No entanto, ele possui algumas peculiaridades, como alta sensibilidade a interferência eletromagnética e dificuldade em capturar o sinal devido a posicionamentos subótimos dos eletrodos

no couro cabeludo. Portanto, um dos aspectos fundamentais que qualquer sistema de processamento de EEG deve ter é a tolerância ao ruído (BIDGOLY; BIDGOLY; AREZOUMAND, 2020).

Os eletrodos *úmidos* são colocados usando gel condutivo e são menos propensos a artefatos provocados por movimentos, que são interferências eletromagnéticas causadas, por exemplo, ao piscar dos olhos. Os eletrodos *secos* não precisam do gel, mas são mais sensíveis a tais artefatos.

O EEG registra as atividades elétricas do cérebro, geralmente colocando eletrodos ao longo da superfície do couro cabeludo. Essas atividades elétricas resultam de fluxos de corrente iônica induzidos pela ativação sináptica sincronizada dos neurônios do cérebro. Elas se manifestam como flutuações de voltagem rítmicas com amplitude variando de 5 a $100\mu V$ e frequência entre 0,5 e 40 Hz (BIDGOLY; BIDGOLY; AREZOUMAND, 2020). As bandas de frequência operacionais no cérebro são as seguintes (SANEI; CHAMBERS, 2021):

- **Delta (1–4Hz)**: A onda mais lenta e geralmente a de maior amplitude. A banda Delta é observada em bebês e durante o sono profundo em adultos.

- **Theta (4–8Hz)**: Observada em crianças, adultos sonolentos e durante a recordação de memórias. A amplitude da onda Theta é tipicamente inferior a $100\mu V$.

- **Alpha (8–12Hz)**: Geralmente a banda de frequência dominante, aparecendo durante a consciência relaxada ou quando os olhos estão fechados. A atenção focada ou a relaxamento com os olhos abertos reduzem a amplitude da banda Alpha. Essas ondas tem amplitudes normalmente inferiores a 50 μV.

- **Beta (12–25Hz)**: Associada ao pensamento, concentração ativa e atenção focada. A amplitude da onda Beta é normalmente inferior a 30 μV.

- **Gamma (acima de 25Hz)**: Observada durante o processamento sensorial múltiplo. Os padrões Gamma têm a menor amplitude.

De acordo com (BIDGOLY; BIDGOLY; AREZOUMAND, 2020) e (VICENTE, 2023), para a maioria das tarefas realizadas pelo cérebro, existem regiões associadas a elas, conforme visto na Tabela Tabela 5.

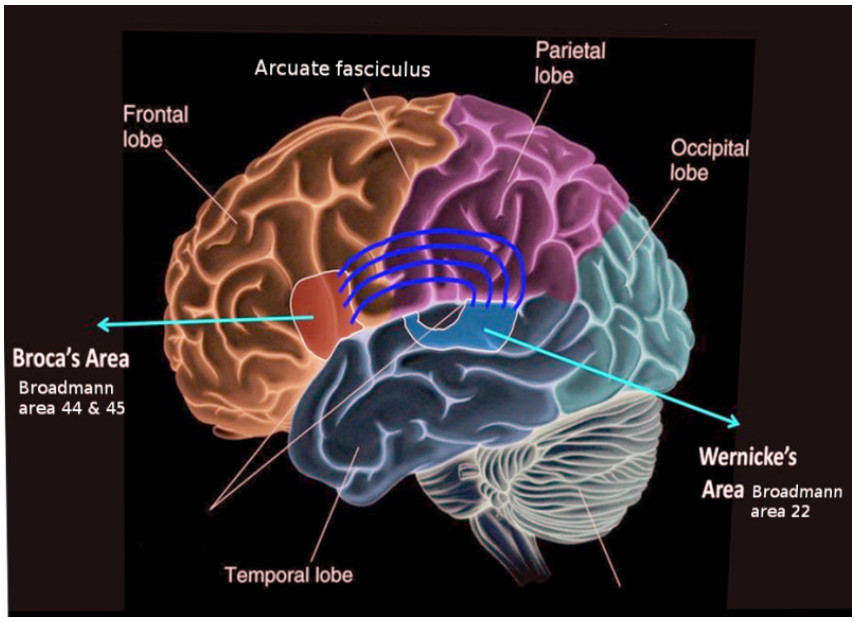
2.1.7 Sistema 10-20 e as áreas do cérebro

Tabela 5 – Tarefas cerebrais e suas regiões correspondentes

Região	Canais	Tarefas
Lobo frontal	Fp1, Fp2, Fpz, Pz, F3, F7, F4, F8	Memória, concentração, emoções.
Lobo parietal	P3, P4, Pz	Resolução de problemas, atenção, sentido do tato.
Lobo temporal	T3, T5, T4, T6	Memória, reconhecimento de faces, audição, palavras e percepção social.
Lobo occipital	O1, O2, Oz	Leitura, visão.
Cerebelo		Controle motor, equilíbrio.
Córtex senso-motor	C3, C4, Cz	Atenção, processamento mental, controle motor fino, integração sensorial.

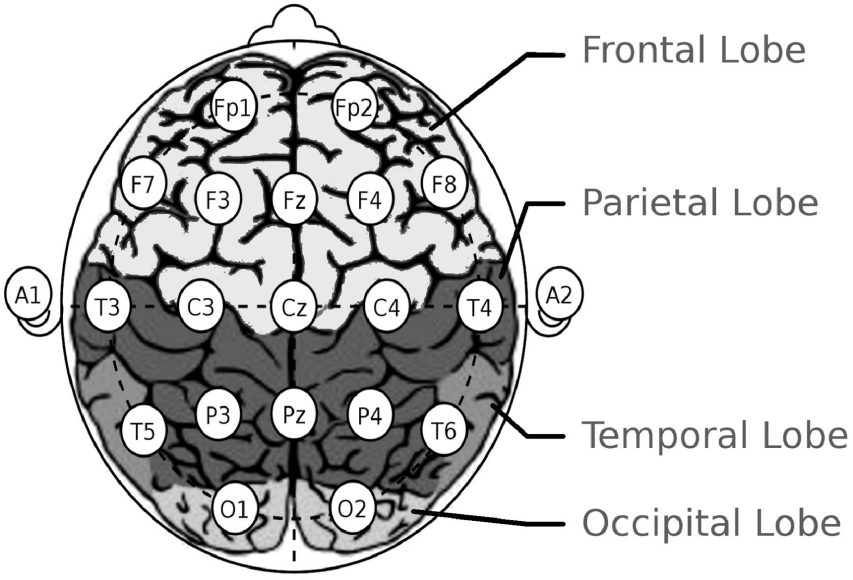
Segundo (ABDULGHANI; WALTERS; ABED, 2023a), essencialmente, a área de Wernicke é responsável por garantir que a fala faça sentido enquanto que a área de broca garante que a fala seja produzida de forma fluente, portanto uma atenção especial deve ser dada a essas regiões. De acordo com a Figura 9 e a Figura 10 os eletrodos frontais e temporais esquerdos F7 e T5 podem estar mais próximos da área de Wernicke e os eletrodos Fp1, F3 e F7 da área de Broca.

Figura 9 – Áreas de Wernicke e Broca



Fonte: (JAVED *et al.*, 2024)

Figura 10 – Posicionamento dos eletrodos de acordo com o padrão 10-20 e lobos cerebrais. Números ímpares são atribuídos aos eletrodos no hemisfério esquerdo, e números pares são atribuídos aos eletrodos no hemisfério direito.



Fonte: (BIDGOLY; BIDGOLY; AREZOUMAND, 2020)

2.1.8 Redes Neurais de Pulso (Spiking Neural Networks)

Redes Neurais (RNs), conforme definido aqui como *uma rede multicamadas, totalmente conectada, com ou sem camadas recorrentes ou convolucionais*, exigem que todos os neurônios sejam ativados tanto na fase de *feed-forward* quanto na de *backpropagation*. Isso implica que cada unidade na rede deve processar alguns dados, resultando em consumo de energia (ESHRAGHIAN *et al.*, 2023).

O sistema sensorial dos sistemas neurológicos biológicos converte dados externos, como luz, odores, toque, sabores e outros, em pulsos. Um pulso é uma alteração na voltagem que é propagada transmitindo informações (KASABOV, 2019). Esses pulsos são então transmitidos ao longo da cadeia neuronal sendo processados, gerando uma resposta ao ambiente.

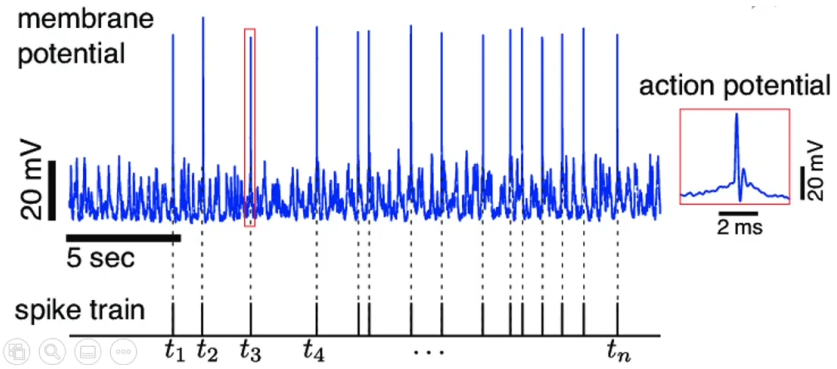
O neurônio biológico dispara apenas quando um certo nível de sinais excitatórios (voltagem) se acumula acima de um limiar em seu citoplasma, permanecendo inativo quando não há sinal, portanto, esse tipo de célula é muito eficiente em termos de consumo de energia e processamento.

Para obter as vantagens mencionadas, as Redes Neurais de Pulso (RNP), em vez de empregar valores de ativação contínuos, como as RNs, utilizam **pulsos** nas camadas de entrada, ocultas e de saída. As RNPs também podem ter entradas contínuas e manter suas propriedades.

Uma RNP **não é** uma simulação um-para-um de neurônios. Em vez disso, ela aproxima certas capacidades computacionais de propriedades biológicas específicas. Alguns estudos, como (JONES; KORDING, 2020), criaram modelos muito mais próximos aos neurônios naturais explorando a não linearidade dos dendritos e outras características neurais, obtendo resultados notáveis na classificação.

Como pode ser visto na Figura 11, os neurônios das RNPs, com os parâmetros corretos, são muito tolerantes a ruídos porque atuam como um **filtro passa-baixa**. Eles geram pulsos mesmo quando um nível considerável de interferência está presente. Tais neurônios são muito sensíveis ao tempo, sendo ótimos para processar fluxos de dados (ESHRAGHIAN *et al.*, 2023).

Figura 11 – Pulsos de um sinal ruidoso.



Fonte: (GOODMAN *et al.*, 2022)

2.1.8.1 Neurônio de Pulso

Embora o foco deste trabalho seja nos *Leaky Integrate and Fire Neurons* (LIF) porque são mais simples, mais eficientes e atualmente generalizam melhor para a maioria dos problemas (GOODMAN *et al.*, 2022), existem modelos mais biologicamente precisos como o **neurônio Hodgkin-Huxley** (GERSTNER *et al.*, 2014) e outros como (JONES; KORDING, 2020) que criaram modelos mais próximos aos neurônios naturais explorando a não linearidade dos dendritos e outras características neurais.

2.1.8.2 Entendendo o LIF

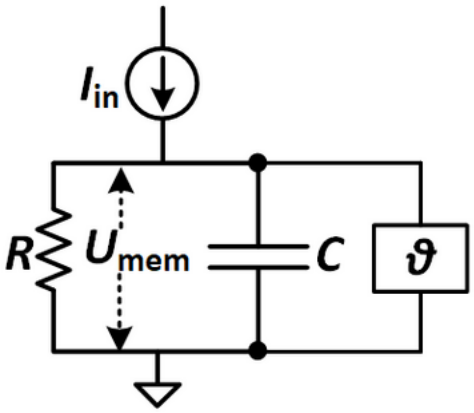
O LIF é um dos modelos de neurônios mais simples em RNPs, ainda assim, pode ser aplicado com sucesso na maioria dos problemas em que as RNPs podem ser usadas.

O LIF, assim como um neurônio de RN, recebe a soma dos inputs ponderados, mas, em vez de passá-lo diretamente para sua função de ativação, algum *vazamento* é aplicado, diminuindo em algum grau a soma.

O LIF se comporta muito como circuitos Resistor-Capacitor, como pode ser visto na Figura 12. Aqui, R é a resistência ao vazamento da corrente, I_{in} é a corrente de entrada,

C é a capacitância, U_{mem} representa é o potencial acumulado e v é um interruptor que permite que o capacitor se descarregue (ou seja, emita um pulso) quando um determinado limiar de potencial é alcançado.

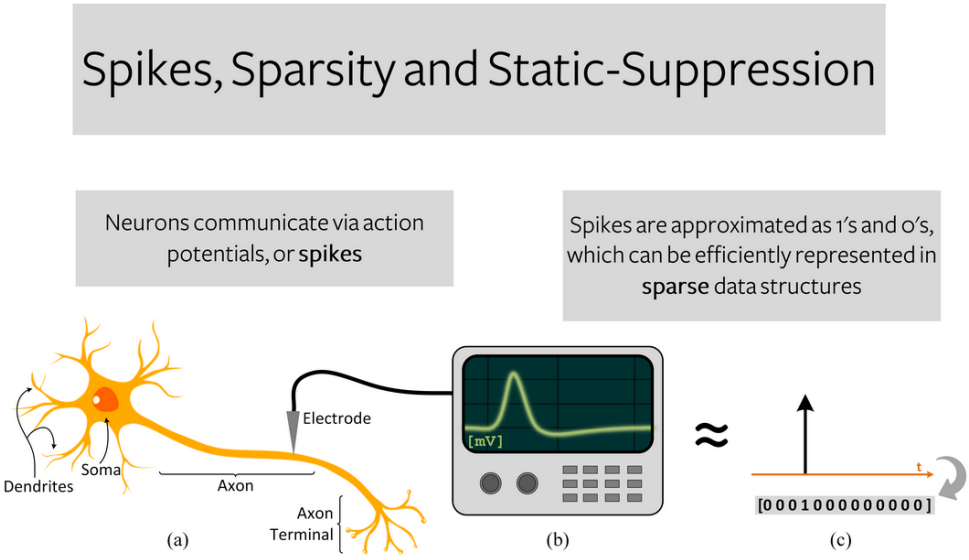
Figura 12 – O modelo RC



Fonte: (ESHRAGHIAN *et al.*, 2023)

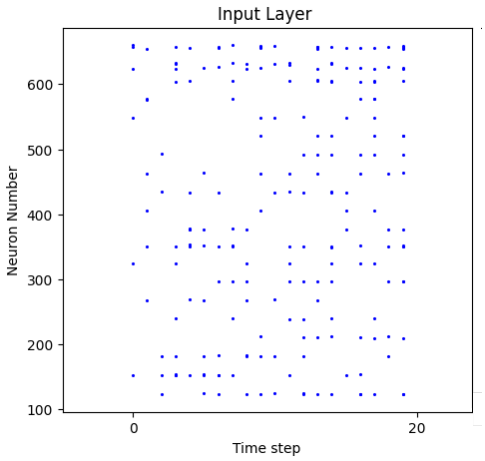
Ao contrário do neurônio Hodgkin-Huxley, os pulsos são representados como **uns** dispersamente distribuídos em uma sequência de **zeros**, como ilustrado nas s Figura 13 e Figura 14. Essa abordagem simplifica os modelos e reduz a potência computacional e o armazenamento necessário para executar uma RNP.

Figura 13 – Dispersão em Redes Neurais de Pulsos.



Fonte: (ESHRAGHIAN *et al.*, 2023)

Figura 14 – Atividade dispersa de uma RNP: O eixo horizontal representa o momento no qual os dados estão sendo processados e o vertical representa o número de neurônios da RNP. Note que, na maior parte do tempo, muito poucos neurônios são ativados.



Fonte: O autor.

Como resultado do mencionado acima, em RNP, a informação é codificada no formato de *tempo* e/ou *taxa* de pulsos, proporcionando consequentemente grandes capacidades de processamento de fluxos de dados, mas limitando o processamento de dados estáticos.

O modelo LIF é governado pelas equações abaixo (ESHRAGHIAN *et al.*, 2023).

Considerando que Q é uma medida de carga elétrica e $V_{\text{mem}}(t)$ é a diferença de potencial na membrana em um determinado tempo t, então a capacitância do neurônio C é dada pela Equação Equação 2.13.

$$C = \frac{Q}{V_{mem}(t)} \tag{2.13}$$

Então, a carga do neurônio pode ser expressa pela Equação Equação 2.14.

$$Q = C.V_{mem}(t) \tag{2.14}$$

To know how these charge changes according to the time (aka current) we can derivate Q as in Equation Equação 2.15. This expression express the current in the capacitive part of the neuron I_C

Para saber como essa carga muda ao longo do tempo (ou seja, medir a corrente), podemos derivar Q como na Equação Equação 2.15. Essa expressão representa a corrente na parte capacitiva do neurônio I_C.

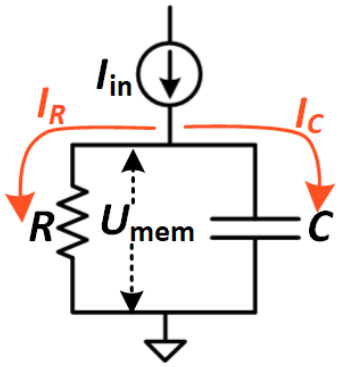
$$I_C = \frac{dQ}{dt} = C.\frac{dV_{mem}(t)}{dt} \tag{2.15}$$

Para calcular a corrente total passando pela parte resistiva do circuito, podemos usar a lei de Ohm:

$$V_{mem}(t) = R.I_R \implies I_R = \frac{V_{mem}(t)}{R} \tag{2.16}$$

Então, considerando que a corrente total não muda, como visto na Figura 15, temos a corrente total de entrada I_{in} do neurônio como na Equação Equação 2.17.

Figura 15 – Modelo RC para correntes: $I_{in} = I_R + I_C$



Fonte: (ESHRAGHIAN *et al.*, 2023)

$$I_{in}(t) = I_R + I_C \implies I_{in}(t) = \frac{V_{mem}(t)}{R} + C.\frac{dV_{mem}(t)}{dt} \tag{2.17}$$

Portanto, para descrever a membrana passiva, temos a Equação linear Equação 2.18.

$$\begin{gathered} I_{in}(t) = \frac{V_{mem}(t)}{R} + C.\frac{dV_{mem}(t)}{dt} \implies \\ I_{in}(t) - \frac{V_{mem}(t)}{R} = C.\frac{dV_{mem}(t)}{dt} \implies \\ \boxed{R.I_{in}(t) - V_{mem}(t) = R.C.\frac{dV_{mem}(t)}{dt}} \end{gathered} \tag{2.18}$$

Então, se considerarmos $\tau = R.C$ como a **constante de tempo da membrana**, obtemos tensões em ambos os lados da Equação Equação 2.19, que **descreve o circuito RC**.

$$
\begin{aligned}
R.I_{in}(t) - V_{mem}(t) &= R.C.\frac{dV_{mem}(t)}{dt} \implies \\
R.I_{in}(t) - V_{mem}(t) &= \tau.\frac{dV_{mem}(t)}{dt} \implies \\
\boxed{\tau.\frac{dV_{mem}(t)}{dt} &= R.I_{in}(t) - V_{mem}(t)}
\end{aligned}
\tag{2.19}
$$

A partir disso, igualando $I_{in} = 0$ (ou seja, sem entrada) e considerando que $\tau = R.C$ é uma constante e, além disso, atribuindo um potencial inicial $V_{mem}(0)$, o comportamento da voltagem do neurônio pode ser modelado como uma curva exponencial, como visto na Equação Equação 2.20.

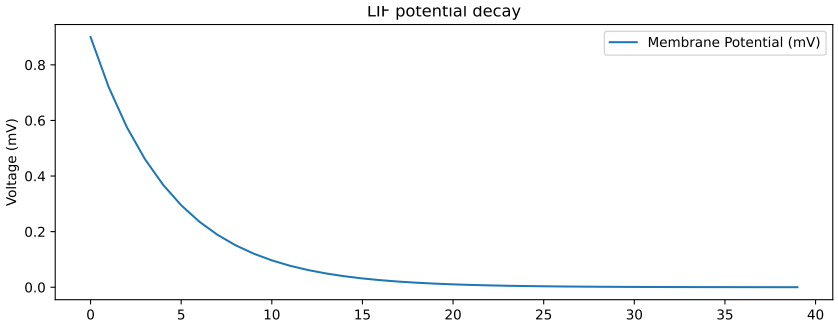
$$
\begin{aligned}
\tau.\frac{dV_{mem}(t)}{dt} &= R.I_{in}(t) - V_{mem}(t) \implies \\
\tau.\frac{dV_{mem}(t)}{dt} &= -V_{mem}(t) = \\
e^{\ln(V_{mem}(t))} &= e^{-\frac{t}{\tau}} = \\
\boxed{V_{mem}(t) &= V_{mem}(0).e^{-\frac{t}{\tau}}}
\end{aligned}
\tag{2.20}
$$

Então, pode-se dizer que: Na ausência de uma entrada I_{in}, o potencial da membrana decai exponencialmente, como ilustrado na Figura 16 e implementado no Código Algoritmo 2.1.

```
def lif(V_mem, dt=1, I_in=0, R=5, C=1):
  tau = R*C
  V_mem = V_mem + (dt/tau)*(-V_mem + I_in*R)
  return V_mem
```

Algoritmo 2.1 – Python implementation of the action potential decaying of a LIF: $I_{in} = 0$

Figura 16 – Decaimento do potencial da membrana.



Fonte: O autor

Com os resultados da Equação Equação 2.19, é possível calcular o aumento da voltagem, conforme visto na Equação Equação 2.21.

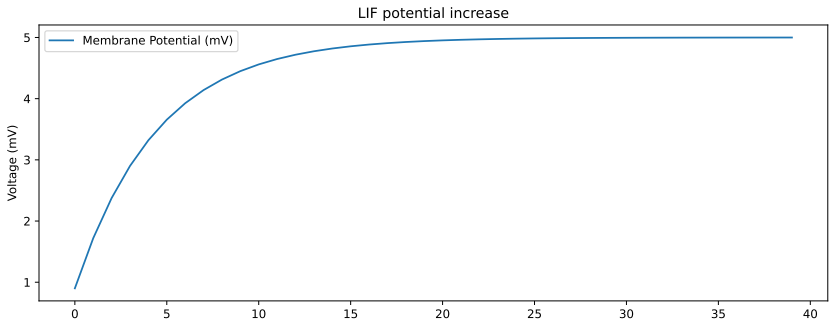
$$
\begin{aligned}
&\tau.\frac{dV_{mem}(t)}{dt} = R.I_{in}(t) - V_{mem}(t) = \\
&\frac{dV_{mem}(t)}{dt} + \frac{V_{mem}(t)}{\tau} = \frac{R.I_{in}(t)}{\tau} \implies \\
&\text{Integrating factor: } e^{\int \frac{1}{\tau}dt} = e^{\frac{1}{\tau}.t} \implies \\
&(e^{\frac{1}{\tau}.t}.V_{mem}(t))' = \frac{R.I_{in}(t)}{\tau}.e^{\frac{1}{\tau}.t} = \\
&\int (e^{\frac{1}{\tau}.t}.V_{mem}(t))' = \int \frac{R.I_{in}(t)}{\tau}.e^{\frac{1}{\tau}.t}dt = \\
&e^{\frac{1}{\tau}.t}.V_{mem}(t) = \int \frac{R.I_{in}(t)}{\tau}.e^{\frac{1}{\tau}.t}dt \therefore \\
&\text{Considering: } V_{mem}(t=0) = 0 \implies \\
&\boxed{V_{mem}(t) = I_{in}(t).R(1 - e^{\frac{1}{\tau}})}
\end{aligned}
\tag{2.21}
$$

Note que quando os potenciais de ação aumentam, ainda há um comportamento exponencial, como visto na Figura 17 e implementado no Código Algoritmo 2.2.

```
1
2 def lif(V_mem, dt=1, I_in=1, R=5, C=1):
3   tau = R*C
4   V_mem = V_mem + (dt/tau)*(-V_mem + I_in*R)
5   return V_mem
```

Algoritmo 2.2 – Python implementation of the action potential decreasing of a LIF: $I_{in} = 1$

Figura 17 – Aumento do potencial da membrana.



Fonte: O autor

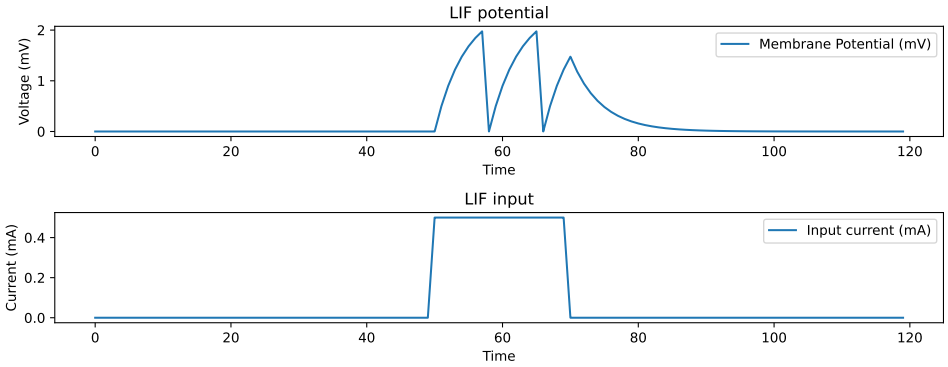
Levando em consideração um certo **limiar** que indica um *reset* nvoltagem do neurônio e dois tipos de *resets* (para zero e subtração do limiar), finalmente é possível modelar o comportamento completo do LIF, conforme ilustrado na Figura 18 e implementado no Código Algoritmo 2.3:

```
1 def lif(V_mem, dt=1, I_in=1, R=5, C=1, V_thresh = 2, reset_zero = True):
```

```
  tau = R*C
  V_mem = V_mem + (dt/tau)*(-V_mem + I_in*R)
  if V_mem > V_thresh:
    if reset_zero:
      V_mem = 0
    else:
      V_mem = V_mem - V_thresh
  return V_mem
```

Algoritmo 2.3 – Python implementation of the action potential full simulation of a LIF: $I_{in} = 1$, $V_{thresh} = 2$ is threshold

Figura 18 – Gráfico do LIF simulado completo: Foram fornecidos 0.5 mA de corrente no intervalo de tempo de 51 a 70.



Fonte: O Autor

2.1.8.3 Outra interpretação do LIF

Começando com a Equação Equação 2.19 e usando o Método de Euler para resolver o modelo LIF:

$$\tau . \frac{dV_{\text{mem}}(t)}{dt} = R.I_{\text{in}}(t) - V_{\text{mem}}(t) \tag{2.22}$$

Resolvendo a derivada na Equação Equação 2.23, obtemos o potencial da membrana para qualquer tempo $t + \Delta t$ no futuro:

$$\begin{aligned} \tau . \frac{V_{\text{mem}}(t+\Delta t) - V_{\text{mem}}(t)}{\Delta t} &= R.I_{\text{in}}(t) - V_{\text{mem}}(t) = \\ V_{\text{mem}}(t+\Delta t) - V_{\text{mem}}(t) &= \frac{\Delta t}{\tau} . (R.I_{\text{in}}(t) - V_{\text{mem}}(t)) = \\ \boxed{V_{\text{mem}}(t+\Delta t) &= V_{\text{mem}}(t) + \frac{\Delta t}{\tau} . (R.I_{\text{in}}(t) - V_{\text{mem}}(t))} \end{aligned} \tag{2.23}$$

2.1.8.4 Treinamento

Como as RNPs são treinadas? Esta ainda é uma questão em aberto. Um neurônio RNP tem o comportamento de sua função de ativação mais parecido com uma **função de passo**. Portanto, em princípio, não podemos usar soluções baseadas em descida de gradiente porque esse tipo de função **não é** diferenciável (KASABOV, 2019).

Mas existem algumas ideias que podem lançar alguma luz sobre este assunto: Enquanto algumas observações *in vivo/in vitro* mostram que os cérebros, em geral, aprendem fortalecendo/enfraquecendo e adicionando/removendo sinapses ou até mesmo criando novos neurônios ou outros métodos complicados como pacotes de RNA, existem algumas mais aceitáveis como as da lista abaixo (KASABOV, 2019):

- **Plasticidade Dependente do Tempo de Pulsos (STDP)**: Se um neurônio pré-sináptico dispara **antes** do pós-sináptico, há um fortalecimento na conexão, mas se o neurônio pós-sináptico disparar antes, então há um enfraquecimento.
- **Descida de Gradiente Emprestada**: Aproxima a função de passo usando outra função matemática, que é diferenciável (como uma sigmoide), para treinar a rede. Essas aproximações são usadas apenas na fase de *backpropagation*, enquanto mantêm a função de passo na fase do *feed-forward* (KASABOV, 2019).
- **Algoritmos Evolutivos**: Usam a seleção dos mais aptos ao longo de muitas gerações de redes.
- **Reservatório/Computação Dinâmica**: **Redes de estado de eco** ou **Máquinas de estado líquido**, respectivamente.

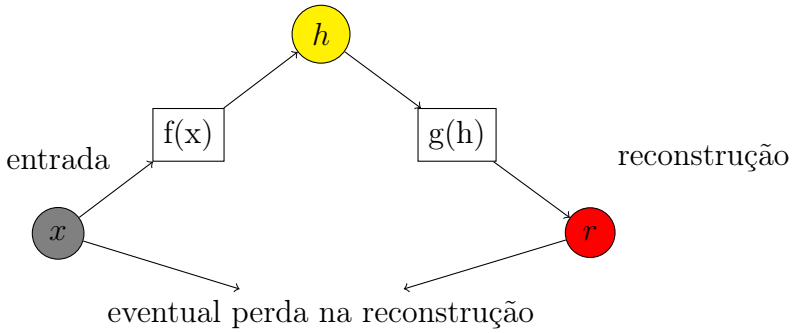
2.1.9 *Autoencoders*

Como ilustrado na Figura 19 *autoencoders* são redes neurais treinadas para reconstruir seus dados de entrada. Elas consistem em uma função codificadora, denotada como $h = f(x)$, e uma função decodificadora que produz uma reconstrução, denotada como $r = g(h)$. A camada oculta h representa um **código** ou representação da entrada que pode ser ou não ser comprimida (GOODFELLOW; BENGIO; COURVILLE, 2016). Tais funções se traduzem em camadas como ilustrado na Figura 20 cujos neurônios são todos unidades ativas com ou sem funções de ativação. É importante destacar que um *autoencoder* pode ser tão raso como o mostrado na figura ou x e r podem ser redes profundas.

O principal objetivo de um *autoencoder* é aprender uma representação dos dados de entrada na camada de **código** e, em seguida, reconstruir os dados de entrada com a maior precisão possível usando o decodificador. Geralmente, os *autoencoders* são projetados

para serem incapazes de copiar perfeitamente os dados de entrada pois isso os tornaria demasiadamente complexos e, em alguns casos, inúteis como será explicado adiante.

Figura 19 – Representação funcional de um *autoencoder*: Sendo o vetor x uma entrada, então o vetor h é resultado da aplicação de uma função codificadora f sobre x: $h = f(x)$, portanto r é a reconstrução de x a partir h através de uma função decodificadora g: $r = g(h)$. Tal processo pode ou não implicar em uma perda de informação.



Fonte: O Autor

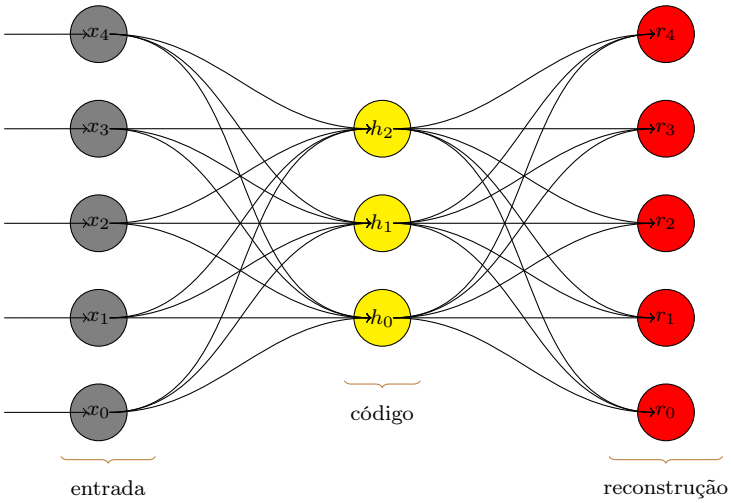
Considerando-se que a reconstrução r seja razoável, isso significa que a região h contém dados suficientes para representar a informação em sua essência, sendo assim, *autoencoders* são ótimos produtores de vetores de características. Hipoteticamente é possível criar *autoencoders* cuja a camada de código seja apenas um número inteiro caso tanto o codificador quanto o decodificador sejam grandes e complexos o suficiente, mas isso se mostraria inútil caso se necessitasse algum tipo de representação da informação original que fosse significativa, ademais, excluindo-se casos triviais, segundo (GOODFELLOW; BENGIO; COURVILLE, 2016) isso ainda não se verifica na prática.

Em se tratando de treinamento as técnicas já conhecidas e testadas para redes neurais em geral (como, por exemplo, *gradient descent* com *minibatch* ou estocástico) podem ser usadas (GOODFELLOW; BENGIO; COURVILLE, 2016) incluindo as formas de regularização com será exposto mais adiante.

2.1.9.1 Autoencoders sub-completos ou clássicos

Esse tipo de rede neural, como ilustrado na Figura 20, tem sua camada de código limitada em dimensionalidade para apenas aproximar a entrada e priorizar certos aspectos dos dados extraindo dessa forma os componentes mais cruciais e significativos de um conjunto de dados (GOODFELLOW; BENGIO; COURVILLE, 2016). Um autoencoder clássico age de forma muito semelhante ao algoritmo *Principal Component Analysis* (PCA) (BENGIO; COURVILLE; VINCENT, 2014).

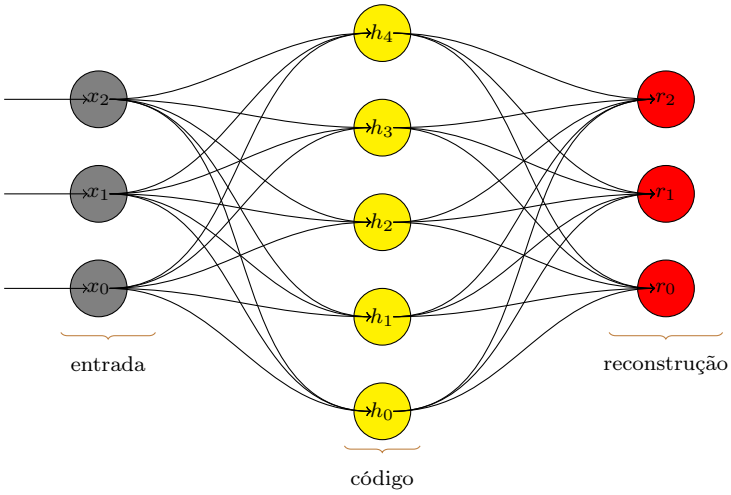
Figura 20 – Exemplo esquemático de um *autoencoder*: Os nós de entrada estão em cinza, a camada de código em amarelo contêm as características codificadas e finalmente a camada de reconstrução em vermelho contêm um cópia aproximada da entrada.



Fonte: O Autor

2.1.9.2 Autoencoders supra-completos

Figura 21 – Exemplo esquemático de um *autoencoder* supra-completo.



Fonte: O Autor

Como mostrado na Figura 21, ao contrário dos autoencoders sub-completos, os supra-completos tem sua camada de código com uma **dimensão maior** do que os dados de entrada. Dessa forma as características dos dados de entrada podem ser codificadas mais de uma vez e uma reconstrução perfeita é atingida. Estruturas assim são inúteis já que não conseguem extrair características significativas a não ser que sejam aplicadas técnicas de regularização (BENGIO; COURVILLE; VINCENT, 2014).

2.1.9.3 *Autoencoders* regularizados

Segundo (BENGIO; COURVILLE; VINCENT, 2014) uma das formas de regularização é justamente a limitação da dimensão da camada de código, a mesma aplicada aos *autoencoders* sub-completos, como estes já foram explicados apresentar-se-ão a seguir outras formas de regularização.

Considerando que *autoencoders* devem ser treinados para minimizar uma função de erro E que compara a entrada original x com a reconstrução r então um autoencoder **regularizado** deve ter adicionado a essa função um termo de regularizador ϕ como a mostrado na Equação Equação 2.24 que é um termo que pode corresponder a uma regularização L1, L2, $\Omega(h)$ (discutidas no apêndice seção E.1).

$$E(x, r) + \phi \tag{2.24}$$

Segundo (GOODFELLOW; BENGIO; COURVILLE, 2016) usando técnicas de regularização é possível até mesmo para um *autoencoder* supra-completo que o mesmo extraia características úteis para uso. Tais técnicas devem prover a seguintes propriedades para a rede:

- dispersão: É a tendência de ativar apenas um pequena quantidade das unidades (também conhecidas como dimensões) da camada de código para qualquer entrada dada criando uma representação mais compacta.
- pequena variação no gradiente: Garantir pouca amplitude nos valores dos gradientes em relação as variações ou ruídos de uma entrada evita que grandes flutuações ocorram na camada de código proporcionando um comportamento estável.

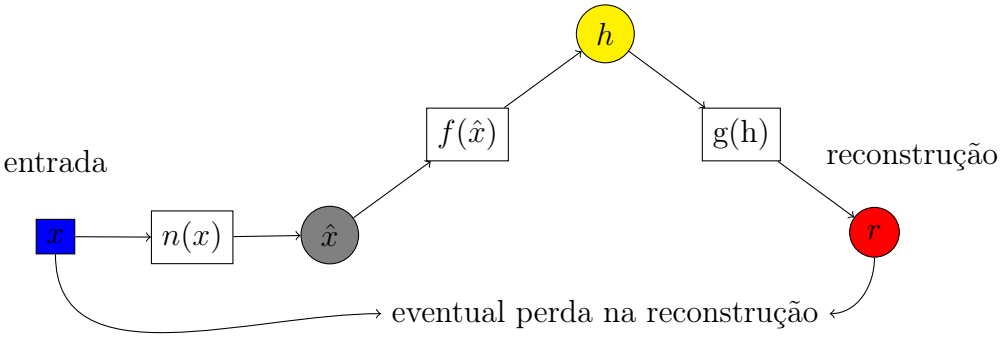
2.1.9.4 *Denoising autoencoders*

Tais redes podem ser caracterizadas também como *autoencoders* regularizados. Porém há uma diferença fundamental: Aqui o foco não é criar funções de regularização e sim adicionar ruídos **apenas** na informação de entrada mas mantendo a intactos os dados que serão comparados na função de erro como mostra a Figura 22. Um *autoencoder* treinado dessa forma, além de ser mais resiliente a ruídos, também adquirire uma funcionalide de preencher falhas de informação nos dados de entrada quando o componente adicionador de ruído é removido e os dados são diretamente apresentados a rede.

2.1.10 Redes neurais residuais (ResNets)

Segundo (HE *et al.*, 2015) a ideia-chave por trás das *ResNets* é a inclusão de conexões de salto como ilustrado na Figura 23, também conhecidas como mapeamentos de

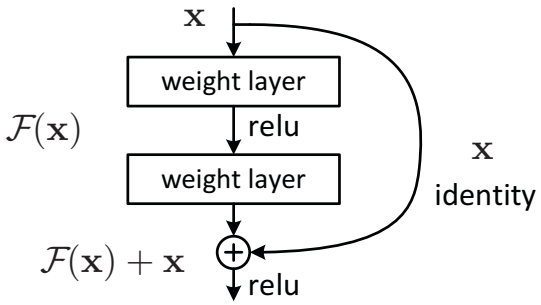
Figura 22 – Representação funcional de um *denoising autoencoder*: Sendo o vetor x uma entrada e n um função que adiciona um ruído aleatório então a informação com ruído $\hat{x}$ é definida como $\hat{x} = n(x)$, portanto o vetor h é resultado da aplicação de uma função codificadora f sobre x: $h = f(x)$, finalmente r é a reconstrução de x a partir h através de uma função decodificadora g: $r = g(h)$. É importante notar que r é comparada com x e não com $\hat{x}$.



Fonte: O autor

identidade, permitindo que a saída de uma camada seja adicionada elemento-a-elemento diretamente à entrada da camada subsequente. Para que isso aconteça é importante que tanto a saída quanto a entrada tenham a mesma dimensão.

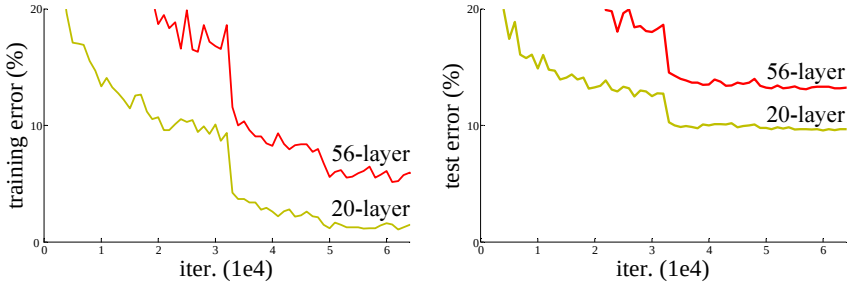
Figura 23 – Bloco Residual: x contorna as camada intermediárias $F(x)$ via uma função identidade somando-se a $F(x)$ ao final do bloco.



Fonte: (HE *et al.*, 2015)

Quando uma rede chega a um certo limite de acurácia, intuitivamente se pode pensar que adicionando mais camadas o desempenho da mesma melhore, no entanto, na prática isso não se verifica. O que se observa em redes mais profundas pode ser exatamente o contrário como mostrado na Figura 24.

Figura 24 – Desempenho de duas redes neurais não residuais com 56 e 20 camadas. A esquerda a taxa de erro durante o treinamento, a direita a taxa de erro em relação ao conjunto de testes.



Fonte: (HE *et al.*, 2015)

Redes neurais muito profundas podem sofrer de problemas como o desaparecimento ou explosão do gradiente, onde os gradientes (ou seja, as direções para ajustar os pesos da rede) se tornam muito pequenos ou grandes à medida que são retro-propagados durante o treinamento. Isso pode dificultar o treinamento eficaz das camadas mais profundas.

Então já que, empiricamente, as redes com menos camadas estão expostas a menos problemas de otimização, se pode juntar várias dessas formando assim uma rede mais profunda e com mais hiperparâmetros que podem ser ajustados. Isso, aliado as técnicas de regularização (ver Apêndice E) diminui a ocorrência de *vanishing/exploding gradients* um problema comum em redes com muitas camadas que pode piorar, impossibilitar ou diminuir a níveis impraticáveis o aprendizado da rede (HE *et al.*, 2015).

Em vez de aprender a mapear diretamente de x para y, as camadas da rede aprendem a mapear de x para a diferença entre x e y, ou seja, para o **resíduo**. Isso simplifica o treinamento, pois a rede precisa apenas ajustar os resíduos, em vez de aprender a mapear completamente as entradas para as saídas.

Além das características já expostas as redes residuais não adicionam complexidade computacional além da negligenciável adição elemento-a-elemento. Isso facilita a comparação com redes não residuais com o mesmo número de hiperparâmetros.

2.1.10.1 Aprendizado residual

Considerando então x como a entrada de uma rede, $f(\cdot)$ como uma série de camadas cujas sucessivas aproximações, via treinamento, resultam em um desejado estado intermediário na camada escondida h, então o que se convenciona chamar de resíduo r é dado na Equação 2.25 considerando que h e x tenha a mesma dimensão.

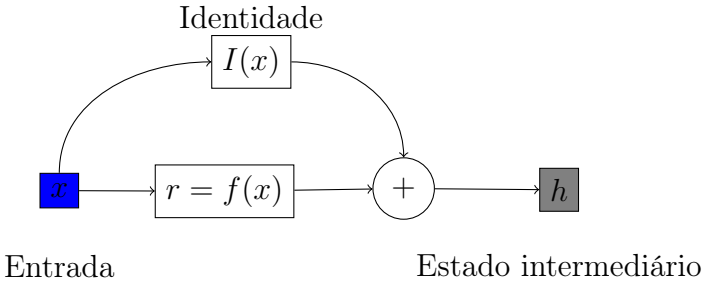
$$r = h - x \tag{2.25}$$

Considerando o resíduo, então se pode reformular o estado h como mostrado na Equação 2.26 demonstrando que, para se mapear x em h basta que $f(\cdot)$ atue sobre r, ou seja, que as aproximações sucessivas sejam feitas sobre o resíduo r.

$$h = r + x \tag{2.26}$$

Se em uma rede não profunda forem adicionadas camadas cuja a única função é receber uma entrada e devolvê-la sem modificações (função identidade) então é trivial que uma rede possa atingir grandes profundidades. No entanto segundo (HE *et al.*, 2015) em redes profundas os otimizadores tem dificuldade em aprender tais funções identidade levando a degradação do desempenho como mostrado na Figura 24. Nas redes residuais se os mapeamentos identidade (também chamados de conexões de atalho) forem ótimos os otimizadores podem simplesmente zerar todos os pesos das camadas intermediárias de um bloco residual.

Figura 25 – Obtenção do resíduo: x um vetor de entrada na rede, $I(\cdot)$ é uma função identidade cujo valor é diretamente somado $\oplus$ com o resultado produzido pelas várias camadas representadas por $f(\cdot)$, h é um vetor representando um estado intermediário de uma rede neural.



Fonte: O autor

O bloco residual mostrado na Figura 25 torna mais fácil para a rede se aproximar de funções identidade pois basta a mesma zerar os pesos constituintes das camadas representadas por $f(\cdot)$. Note que $f(\cdot)$ é flexível e portanto pode ter n camadas constituintes e que essa estrutura pode ser usada multiplas vezes sozinha ou em conjunto com outras estruturas em uma rede.

2.2 Trabalhos mais recentes

Afim de entender melhor o campo de pesquisa, além de formar uma base sólida para compreensão do assunto, foram levantados os métodos de autenticação por fala e/ou análise de sinais cerebrais. A coleta dos trabalhos foi feita segundo os parâmetros descritos abaixo.

2.2.1 Protocolo de coleta

Critérios de inclusão e exclusão de artigos:

- dos últimos 5 anos
- que correspondiam às palavras-chave de pesquisa constantes na Tabela 6
- cujo *abstract* continha as bases de dados usadas, os resultados e a metodologia brevemente descritos.

A **estratégia de busca** dos artigos foi guiada segundo a base de dados e palavras-chaves constantes na Tabela 6.

Tabela 6 – Bases de dados usadas

Base de dados	Palavras chave	Filtros aplicados	Limitações
Web of Science			
ACM			
Elsevier			
IEEE			

Fonte: Elaborado pelo autor, 2024

O primeiro estudo consultado (PARK; LEE, 2023) aborda o uso de EEG para captação de fala imaginada usando as técnicas de decomposição de sinal NA-MEMD (*multivariate empirical mode decomposition*) e DTWPT (Discrete time Wavelet Package Transform) e a arquitetura de redes neurais MRF-CNN. A NA-MEMD é uma extensão do EMD (*Empirical mode decomposition*) para sinais multivariados permitindo a decomposição de sinais. O MRF-CNN é uma arquitetura de rede neural convolucional profunda que utiliza características estatísticas extraídas dos sinais decompostos como entrada para o classificador. Foram utilizadas técnicas de validação cruzada e divisão de dados para treinamento e teste do classificador. Os dados EEG foram coletados dos nove participantes saudáveis, estes tiveram que imaginar as vogais "a", "e", "i", "o", "u". O estudo atingiu uma média de 80,41% de classificações corretas.

Em (ABDULGHANI; WALTERS; ABED, 2023b) usa *Wavelet Scattering Transform*, técnica que aplica sobre o sinal transformadas wavelet combinadas operações não lineares, para extração de características para posterior classificação por uma Rede Neural LSTM(*Long-Short term memory*) recorrente. Os quatro participantes imaginaram as seguintes expressões: *"up", "down", "left"e "right"*. Em testes a acurácia alcançada foi 92,50%.

Com o fim de remover ruídos e artefatos causados por movimentos musculares relacionados a região da cabeça em (MAHAPATRA; BHUYAN, 2023) foi usada a DTWT

(*Discrete Time Wavelet Transform*) para processar os sinais EEG, que foram obtidos a partir das bases de dados EPOC e MUSES (VIVANCOS, 2023). Depois da filtragem e geração de características variações de Redes Neurais Recorrentes Bidirecionais foram empregadas na classificação das classes compostas pelos dígitos de 0 a 9. Foram alcançadas acurácias máximas de 98,18% na base MUSE e 71,60% na EPOC.

Nesta revisão (SHAH *et al.*, 2022) que foi feita segundo as orientações constantes no protocolo PRISMA-ScR foram selecionados 34 trabalhos que analisaram palavras e expressões imaginadas, nestes, a técnica mais usada para extração de características foram os filtros Wavelet e os algoritmos de classificação mais proeminentes foram as *Support Vector Machines*(SVM) e as Redes Neurais Convolucionais. É reportado os estudos utilizam poucos tipos de métricas limitando-se na maioria dos casos a acurácia, a revisão sugere que também sejam usadas métricas como: Precisão, *Recall*, Taxa de erro de palavras (Word Error Rate), AUC (Área Under the Curve), Sensitividade, Especificidade e Média Quadrática afim de se oferecer uma visão mais geral dos resultados.

Uma combinação de redes neurais convolucionais temporais (TCN) e convolucionais tradicionais (CNN) foi usada no estudo (MAHAPATRA; BHUYAN, 2022) para classificar sinais de EEG pré-processados pelo algoritmo *FastICA* afim de proporcionar uma melhor separação dos canais e por DTWT para remoção de artefatos e criação dos vetores de características. Os sinais foram obtidos da base (CORETTO; GAREIS; RUFINER, 2017) que é composta por sinais produzidos por 15 indivíduos falantes de espanhol com as vogais "a", "e", "i", "o", "u"e com os comandos *"arriba", "abajo", "derecha"e "isquierda"*. O estudo alcançou uma acurácia de 96,49%.

Para classificar as letras do alfabeto imaginadas em inglês por 13 indivíduos o estudo (AGARWAL; KUMAR, 2022) usou a DTWT para remoção ruídos e separação de bandas de frequência do sinal a ser classificado por dois algoritmos simultaneamente: Árvores Aleatórias (ou *Random Forrest*)(RF) e SVM. O estudo alcançou uma acurácia média entre as classificações de todos os elementos do alfabeto de 77,97%. Neste trabalho também foi utilizado o método *Common Average Referecing*(CAR) que, via a subtração de uma média dos sinais captados em um certo intervalo de tempo, procura remover interferências induzidas em todos os eletrodos por variáveis ambientais aumentando a relação sinal/interferência (*Signal-to-noise ratio*)(SNR). Uma interessante constatação desse trabalho é de que as ondas cerebrais α, β e θ quando consideradas sozinhas melhoram o desempenho dos classificadores.

Aqui (HERNáNDEZ-DEL-TORO; REYES-GARCíA; VILLASEñOR-PINEDA, 2021) usa cinco métodos de extração de características: DTWT (para obtenção da energia de bandas), EMD, dimensões fractais e características extraídas segundo metodos da teoria do caos. Os dados foram obtidos a fim de se criar 3 bases de dados contendo as palavras em espanhol *"arriba", "abajo", "derecha"e "isquierda"*, a primeira continha in-

formações de 27 pessoas, a segunda também 27 e a terceira 20. Na primeira e segunda bases a taxa de captura foi de 128Hz e na terceira 500hz, esta foi subamostrada para 128Hz a posteriori. Para remoção de ruído o método CAR foi usado. Os classificadores usados foram RF, *K-nearest-neighbors*(KNN), SVM e Regressão Logística. Para avaliação as métricas Pontuação F1 (PF1), Precisão e *Recall* foram usadas. Os melhores resultados obtidos nas bases 1, 2 e 3 foram respectivamente: 0,73, 0,79 e 0,68 para PF1, 0,66, 0,70, 0.65 para Precisão e 0,87, 0,93 e 0,83 para *Recall*.

Constante nas referências de (HERNáNDEZ-DEL-TORO; REYES-GARCíA; VILLASEñOR-PINEDA, 2021) um dos poucos trabalhos que discorreram sobre a identificação de pessoas (MOCTEZUMA *et al.*, 2019) usou CAR para melhorar a SNR dos sinais armazenados e amostrados a 128Hz correspondentes as palavras em espanhol *"arriba", "abajo", "derecha"e "isquierda"* imaginadas por 27 voluntários. Foi tratada também a identificação independentemente da palavra imaginada. A geração do vetor de características foi feita usando DTWT em 4 níveis e para cada nível foi calculada a Energia de *Teager* ou *Teager Energy Operator* (TEO). RF foi usada como classificador. Os resultados variaram entre 85% quando todos os 27 indivíduos foram incluídos até 97% quando apenas 5 foram considerados.

Dada a pouca quantidade de dados e bases existentes quando se trata de EEG, o artigo (PANACHAKEL; RAMAKRISHNAN, 2021) usou uma técnica de expansão de informação baseada em "janelas deslizantes"ao mesmo tempo em que, em vez de considerar separadamente cada sensor, agrupou os dados dos mesmos em matrizes tridimensionais semelhantes a imagens que foram processadas por uma rede Resnet50 (HE *et al.*, 2015) adaptada usando a técnica de transferência de aprendizado. A base de dados usada (NGUYEN; KARAVAS; ARTEMIADIS, 2018) é composta por EEG capturado de 64 canais a 1000Hz para as seguintes falas imaginadas por 11 pessoas do sexo masculino e 4 do feminino: *"independent", "cooperate", "in", "out", "up", "a", "i", "u"*. As taxas de amostragem foram posteriormente diminuídas para 250Hz e filtradas para remoção de ruídos e artefatos. Os resultados variaram de um mínimo de 79,7% de acurácia para classificação de vogais até um máximo de 95,5% para palavra curtas e longas.

(TAMM; MUHAMMAD; MUHAMMAD, 2020) usou uma base de dados composta por 15 voluntários imaginando as letras *"a", "e", "i", "o", "u"* e mais seis diferentes palavras (CORETTO; GAREIS; RUFINER, 2017) e simplificou a estrutura de uma CNN criada em outro estudo usando uma técnica de transferência de aprendizado enquanto tentou manter a mesma acurácia. Os dados iniciais foram submetidos a sub-amostragem para que se chegasse a uma taxa de amostragem de 128Hz e dispostos em uma matriz de duas dimensões. O classificador simplificado atingiu uma acurácia média de 23,98%.

(PANACHAKEL; RAMAKRISHNAN; ANANTHAPADMANABHA, 2019) utilizou 11 palavras da base de dados (ZHAO; RUDZICZ, 2015), extraindo de cada canal

características baseadas em transformadas *Wavelet* de nível 7, alcançando uma acurácia média de 57,15%. A inovação aqui foi tratar separadamente os vetores de características dos 11 canais, inserindo-os, um por vez, em uma rede neural profunda com 2 camadas ocultas. A classificação foi realizada por votação de maioria simples após o processamento dos 11 sinais da fala imaginada a ser classificada. A base de dados é composta por 7 fonemas e 4 palavras imaginadas, com uma taxa de amostragem de 1KHz. Os dados foram filtrados em uma banda de 1 a 50Hz.

Usando uma rede neural profunda de 4 camadas ocultas como classificador para duas palavras ("in"e "cooperate") presentes no banco de dados referenciado em (DASALLA *et al.*, 2009), (PANACHAKEL; RAMAKRISHNAN; ANANTHAPADMANABHA, 2020) criou vetores de características usando a transformada wavelet de 4 níveis, extraindo características como variância, entropia e Root Mean Square (RMS). Esses vetores foram então classificados usando um sistema de votação, onde cada vetor de características de cada canal é avaliado individualmente e a classe é determinada por maioria simples (similar a (PANACHAKEL; RAMAKRISHNAN; ANANTHAPADMANABHA, 2019)). O banco de dados é composto por amostras coletadas de 15 indivíduos com taxa de amostragem de 1KHz. O sistema obteve uma acurácia média de 71,8%.

Apesar da autenticação por voz não ser um tema novo, há poucos trabalhos recentes que tratam do assunto. No que concerne a locutores disfônicos arrisca-se dizer que não há trabalhos recentes relacionados. Dito isso (WANG *et al.*, 2023) combinou a arquitetura Res2Net (GAO *et al.*, 2021) com *Perceptual Wavelet Packet Entropy* (PWPE) e um bloco *Efficient Channel Attention* (ECA) (WANG *et al.*, 2019) para reconhecer indivíduos cujas vozes constam da base de dados (FAN *et al.*, 2020) composta por vozes de celebridades chinesas em várias situações (gravação em estúdio, palestras, gravações em ambientes ruidosos, etc.). PWPE consiste em "podar"os nós da árvore de decomposição *wavelet package* segundo seus "centros de frequência"e energia definidos anteriormente de acordo com a amplitude de audição da espécie (no caso: Humanos), dessa forma apenas as frequências pertencentes à escala auditiva humana permanecem. Em seguida é aplicado o operador não normalizado de entropia de Shannon gerando assim os vetores de características. Nesse artigo o objetivo foi verificar como os modelos se comportariam em termos de EER se fossem treinados em um contexto e depois aplicados a outros com, por exemplo, treinar em uma palestra em um auditório e testar com dados vindos de ambientes externos. Os resultados de classificação foram, em média, de menos do que 8% de EER para contextos de médio e alto ruído.

Conclusão: Os estudos consultados forneceram ideias para a construção do protocolo para coleta de dados que será explicado no Capítulo 3, além disso, trouxeram a importância da filtragem das frequências próprias da rede elétrica local (MAHAPATRA; BHUYAN, 2023) afim de que tais interferências não sejam consideradas no vetor

de características. Ainda no tópico de captação dos sinais (MAHAPATRA; BHUYAN, 2022) usou o algoritmo *FastICA* afim de garantir uma melhor separabilidade entre os vários canais EEG. Em um trabalho anterior (ANDRé, 2021) e na revisão (SHAH *et al.*, 2022) se demonstrou que, em se tratando de processamento de sinais, a fase de extração de características é de grande importância pois esta impacta de forma sensível no desempenho dos algoritmos de classificação. Também em (SHAH *et al.*, 2022) foram destacadas várias métricas que contribuirão para uma melhor avaliação dos resultados nos procedimentos futuros, além disso, (AGARWAL; KUMAR, 2022) lembrou a importância de separar indivíduos por sexo, destros e canhotos além de constatar que as ondas cerebrais α, β e θ quando consideradas sozinhas melhoram o desempenho dos classificadores. Em se tratando de tratamento dos sinais além da DTWT, CAR pareceu uma escolha bem comum entre os trabalhos (AGARWAL; KUMAR, 2022), (HERNáNDEZ-DEL-TORO; REYES-GARCíA; VILLASEñOR-PINEDA, 2021), (MOCTEZUMA *et al.*, 2019). Fez muito sentido quem em (PANACHAKEL; RAMAKRISHNAN, 2021) os sinais foram analisados de forma conjunta criando matrizes tridimensionais, pois a fala imaginada acontece em todo o cérebro, sendo assim, analisar de forma isolada cada canal pode prejudicar o aprendizado das correlações entre os sinais. Em (TAMM; MUHAMMAD; MUHAMMAD, 2020) foi possível constatar que nem sempre redes neurais mais profundas terão um desempenho muito melhor do redes mais rasas. Indo na contramão da maioria dos estudos (PANACHAKEL; RAMAKRISHNAN; ANANTHAPADMANABHA, 2019) e (PANACHAKEL; RAMAKRISHNAN; ANANTHAPADMANABHA, 2020) trouxeram uma ideia interessante: Como os sinais da fala imaginada são altamente correlacionados, uma forma de aumentar a quantidade de dados fornecidos ao classificador é considerar separadamente cada um dos canais de EEG como uma entrada à rede de forma que, ao fim da classificação de todos os canais, a classe mais votada seja o resultado da classificação. E se tratando de transformadas Wavelet (WANG *et al.*, 2023) trouxe uma ideia interessante: Selecionar os nós de uma árvore de decomposição Wavelet-Packet segundo uma amplitude de frequências desejadas, no caso, aquelas pertencentes a voz humana, no entanto, é interessante destacar que tal técnica pode ser aplicada a qualquer sinal, o que pode ser muito útil no momento do tratamento do sinal tanto de voz quando de EEG.

3 Abordagem proposta

3.1 A Base de sinais

3.1.1 Coleta dos sinais

3.1.2 Organização da base de sinais

3.2 Estrutura da estratégia proposta

3.3 Procedimentos

3.3.1 Protocolo de Coleta de Dados

O procedimento tem como objetivo a coleta de uma ampla gama de dados, começando com a gravação exclusiva das falas imaginadas. Posteriormente, serão registradas as falas pronunciadas. Esse método permitirá a obtenção de dados sobre a fala imaginada sem a interferência dos músculos relacionados à fala, além dos dados sobre a fala efetivamente pronunciada.

1. Primeiramente será exibido ao participante um sinal visual de espera por 5 segundos.
2. Executar o ciclo silencioso.
    a) Por 5 segundos é exibida a palavra que deve ser pensada.
    b) Por 5 segundos a palavra deve ser **imaginada** uma única vez.
    c) Um sinal visual é emitido novamente por 5 segundos
    d) Por 5 segundos a palavra deve ser **pronunciada** uma única vez.
3. Executar o ciclo ruidoso.
    a) Por 5 segundos é exibida a **mesma** palavra que deve ser pensada e sons de ambientes ruidosos começam a ser reproduzidos.
    b) Por 5 segundos a palavra deve ser **imaginada** uma única vez.
    c) Um sinal visual é emitido novamente por 5 segundos
    d) Por 5 segundos a palavra deve ser **pronunciada** uma única vez.
    e) A reprodução do ruído é finalizada.
4. Se deve selecionar uma nova palavra e voltar para o Item 2.

3.3.2 Tratamento do sinal

Afim de evitar as interferências intrínsecas à captação do sinal o mesmo deve passar por um **filtro passa baixa de 100Hz** mantendo por uma boa margem as frequências operacionais típicas de um cérebro humano (BIDGOLY; BIDGOLY; AREZOUMAND, 2020).

3.3.3 Procedimento 01

4 Testes e Resultados

4.1 Procedimento 01

5 Conclusões e Trabalhos Futuros

Referências

ABDULGHANI, M. M.; WALTERS, W. L.; ABED, K. H. Imagined speech classification using eeg and deep learning. *Bioengineering*, v. 10, n. 6, 2023. ISSN 2306-5354. Disponível em: <https://www.mdpi.com/2306-5354/10/6/649>.

ABDULGHANI, M. M.; WALTERS, W. L.; ABED, K. H. Imagined speech classification using eeg and deep learning. *Bioengineering (Basel)*, MDPI AG, Switzerland, v. 10, n. 6, p. 649–, 2023. ISSN 2306-5354.

ADDISON, P. S.; WALKER, J.; GUIDO, R. C. Time–frequency analysis of biosignals. *IEEE Engineering in Medicine and Biology Magazine*, v. 28, n. 5, p. 14–29, Sep 2009.

AGARWAL, P.; KUMAR, S. Electroencephalography based imagined alphabets classification using spatial and time-domain features. *International journal of imaging systems and technology*, John Wiley and Sons, Inc, Hoboken, USA, v. 32, n. 1, p. 111–122, 2022. ISSN 0899-9457.

ANDRé, F. *Caracterização de voice spoofing para fins de verificação de locutores com base na Transformada Wavelet e na Análise Paraconsistente de Características*. Dissertação (Dissertação de Mestrado) — Universidade Estadual Paulista - campus de São José do Rio Preto-SP, São José do Rio Preto, Brazil, 2021. Orientador: Prof Dr Rodrigo Capobianco Guido.

BENGIO, Y.; COURVILLE, A.; VINCENT, P. *Representation Learning: A Review and New Perspectives*. 2014.

BERANEK, L. L. *Acoustic Measurements*. [S.l.]: J. Wiley, 1949.

BIANCHI, G. *Electronic Filter Simulation & Design*. [S.l.]: McGraw-Hill Education, 2007. ISBN 9780071712620.

BIDGOLY, A. J.; BIDGOLY, H. J.; AREZOUMAND, Z. A survey on methods and challenges in eeg based authentication. *Computers and Security*, v. 93, p. 101788, 2020. ISSN 0167-4048. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167404820300730>.

BOSI, M.; GOLDBERG, R. E. *Introduction to digital audio coding and standards*. [S.l.]: Springer Science & Business Media, 2002. v. 721.

BUTTERWORTH, S. On the theory of filters amplifiers. 1930.

CORETTO, G. A. P.; GAREIS, I. E.; RUFINER, H. L. Open access database of eeg signals recorded during imagined speech. In: ROMERO, E.; LEPORE, N.; BRIEVA, J.; BRIEVA, J.; AND, I. L. (Ed.). *12th International Symposium on Medical Information Processing and Analysis*. SPIE, 2017. v. 10160, p. 1016002. Disponível em: <https://doi.org/10.1117/12.2255697>.

COSTA, N. C. d.; ABE, J. M. Paraconsistência em informática e inteligência artificial. *Estudos Avançados*, scielo, v. 14, p. 161 – 174, 08 2000.

COSTA, N. da; BÉZIAU, J.; BUENO, O. *Elementos de teoria paraconsistente de conjuntos*. Centro de Lógica, Epistemologia e História da Ciência, UNICAMP, 1998. (Coleção CLE). Disponível em: <https://books.google.com.br/books?id=MGCKGwAACAAJ>.

DASALLA, C. S.; KAMBARA, H.; SATO, M.; KOIKE, Y. Single-trial classification of vowel speech imagery using common spatial patterns. *Neural networks : the official journal of the International Neural Network Society*, Elsevier, v. 22, n. 9, p. 1334–1339, 2009.

DAUBECHIES, I. *Ten Lectures on Wavelets*. [S.l.]: Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 1992. (CBMS-NSF Regional Conference Series in Applied Mathematics).

ESHRAGHIAN, J. K.; WARD, M.; NEFTCI, E. O.; WANG, X.; LENZ, G.; DWIVEDI, G.; BENNAMOUN, M.; JEONG, D. S.; LU, W. D. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, v. 111, n. 9, p. 1016–1054, 2023.

FAN, Y.; KANG, J.; LI, L.; LI, K.; CHEN, H.; CHENG, S.; ZHANG, P.; ZHOU, Z.; CAI, Y.; WANG, D. Cn-celeb: a challenging chinese speaker recognition dataset. In: IEEE. *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.], 2020. p. 7604–7608.

FREITAS, S. Avaliação acústica e áudio percetiva na caracterização da voz humana. Faculdade de Engenharia da Universidade do Porto, 2013.

GAO, S.-H.; CHENG, M.-M.; ZHAO, K.; ZHANG, X.-Y.; YANG, M.-H.; TORR, P. Res2net: A new multi-scale backbone architecture. *IEEE TPAMI*, 2021.

GERSTNER, W.; KISTLER, W.; NAUD, R.; PANINSKI, L. *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, 2014. ISBN 9781107060838. Disponível em: <https://neuronaldynamics.epfl.ch/online/Ch2.S2.html>.

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

GOODMAN, D.; FIERS, T.; GAO, R.; GHOSH, M.; PEREZ, N. *Spiking Neural Network Models in Neuroscience - Cosyne Tutorial 2022*. Zenodo, 2022. Disponível em: <https://doi.org/10.5281/zenodo.7044500>.

GUIDO, R. C. Practical and useful tips on discrete wavelet transforms [sp tips tricks]. *IEEE Signal Processing Magazine*, v. 32, n. 3, p. 162–166, May 2015.

GUIDO, R. C. Paraconsistent feature engineering [lecture notes]. *IEEE Signal Processing Magazine*, v. 36, n. 1, p. 154–158, Jan 2019.

HAYKIN, S. *Redes Neurais: Princípios e Prática*. [S.l.]: Bookman Editora, 2001. ISBN 9788577800865.

HAYKIN, S.; MOHER, M. *Sistemas de Comunicação-5*. [S.l.]: Bookman Editora, 2011.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. Disponível em: <http://arxiv.org/abs/1512.03385>.

HERNáNDEZ-DEL-TORO, T.; REYES-GARCíA, C. A.; VILLASEñOR-PINEDA, L. Toward asynchronous eeg-based bci: Detecting imagined words segments in continuous eeg signals. *arXiv.org*, Cornell University Library, arXiv.org, Ithaca, 2021. ISSN 2331-8422.

JAVED, K.; REDDY, V.; DAS, M.; AL. et. Neuroanatomy, wernicke area. *StatPearls [Internet]*, StatPearls Publishing, Treasure Island (FL), Jan 2024. [Updated 2023 Jul 24]. Available from: <https://www.ncbi.nlm.nih.gov/books/NBK533001/>.

JENSEN, A.; COUR-HARBO, A. la. *Ripples in Mathematics*. [S.l.]: Springer Berlin Heidelberg, 2001.

JOLLIFFE, I. *Principal Component Analysis*. Springer, 2002. (Springer Series in Statistics). ISBN 9780387954424. Disponível em: <https://books.google.com.br/books?id=_olByCrhjwIC>.

JONES, I. S.; KORDING, K. P. *Can Single Neurons Solve MNIST? The Computational Power of Biological Dendritic Trees*. 2020. Disponível em: <https://arxiv.org/abs/2009.01269>.

KASABOV, N. K. *Time-space, spiking neural networks and brain-inspired artificial intelligence*. [S.l.]: Springer, 2019.

KREMER, R. L.; GOMES, M. d. C. A eficiência do disfarce em vozes femininas: uma análise da frequência fundamental. *ReVEL*, v. 12, p. 23, 2014.

MAHAPATRA, N. C.; BHUYAN, P. Multiclass classification of imagined speech vowels and words of electroencephalography signals using deep learning. *Advances in human-computer interaction*, Hindawi, New York, v. 2022, p. 1–10, 2022. ISSN 1687-5893.

MAHAPATRA, N. C.; BHUYAN, P. Eeg-based classification of imagined digits using a recurrent neural network. *Journal of neural engineering*, IOP Publishing, England, v. 20, n. 2, p. 26040–, 2023. ISSN 1741-2560.

MOCTEZUMA, L. A.; TORRES-GARCíA, A. A.; VILLASEñOR-PINEDA, L.; CARRILLO, M. Subjects identification using eeg-recorded imagined speech. *Expert Systems with Applications*, v. 118, p. 201–208, 2019. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417418306468>.

NGUYEN, C. H.; KARAVAS, G. K.; ARTEMIADIS, P. Inferring imagined speech using eeg signals: a new approach using riemannian manifold features. *Journal of neural engineering*, v. 15, n. 1, p. 016002, 2018. Disponível em: <https://doi.org/10.1088/1741-2552/aa8235>.

PANACHAKEL, J.; RAMAKRISHNAN, A. Decoding imagined speech from eeg using transfer learning. *IEEE Access*, PP, 10 2021.

PANACHAKEL, J. T.; RAMAKRISHNAN, A.; ANANTHAPADMANABHA, T. Decoding imagined speech using wavelet features and deep neural networks. In: *2019 IEEE 16th India Council International Conference (INDICON)*. IEEE, 2019. Disponível em: <http://dx.doi.org/10.1109/INDICON47234.2019.9028925>.

PANACHAKEL, J. T.; RAMAKRISHNAN, A. G.; ANANTHAPADMANABHA, T. V. *A Novel Deep Learning Architecture for Decoding Imagined Speech from EEG*. 2020.

PARK, H.-j.; LEE, B. Multiclass classification of imagined speech eeg using noise-assisted multivariate empirical mode decomposition and multireceptive field convolutional neural network. *Frontiers in human neuroscience*, Frontiers Media S.A, v. 17, p. 1186594–1186594, 2023. ISSN 1662-5161.

POLIKAR, R. *et al. The wavelet tutorial*. 1996.

RASHID, T. *Make Your Own Neural Network: A Gentle Journey Through the Mathematics of Neural Networks, and Making Your Own Using the Python Computer Language*. [S.l.]: CreateSpace Independent Publishing, 2016. ISBN 9781530826605.

SALOMON, D.; MOTTA, G.; BRYANT, D. *Data Compression: The Complete Reference*. Springer London, 2007. (Molecular biology intelligence unit). ISBN 9781846286032. Disponível em: <https://books.google.com.br/books?id=ujnQogzx_2EC>.

SANEI, S.; CHAMBERS, J. A. *EEG signal processing and machine learning*. [S.l.]: John Wiley & Sons, 2021.

SHAH, U.; ALZUBAIDI, M.; MOHSEN, F.; ABD-ALRAZAQ, A.; ALAM, T.; HOUSEH, M. The role of artificial intelligence in decoding speech from eeg signals: A scoping review. *Sensors (Basel, Switzerland)*, MDPI AG, Basel, v. 22, n. 18, p. 6975–, 2022. ISSN 1424-8220.

TAMM, M.-O.; MUHAMMAD, Y.; MUHAMMAD, N. Classification of vowels from imagined speech with convolutional neural networks. *Computers*, MDPI, v. 9, n. 2, p. 46, 2020.

VALENÇA, E. H. O. *et al.* Análise acústica dos formantes em indivíduos com deficiência isolada do hormônio do crescimento. Universidade Federal de Sergipe, 2014.

VICENTE, E. *Sistema 10-20 para localizar alvos terapêuticos em EMT*. 2023. Disponível em: <https://www.kandel.com.br/post/como-localizar-os-pontos-de-estimulacao-para-estimulacao-magnetica-transcraniana>.

VIVANCOS, D. *MindBigData*. 2023. Disponível em: <https://mindbigdata.com/opendb/>.

WANG, Q.; WU, B.; ZHU, P.; LI, P.; ZUO, W.; HU, Q. Eca-net: Efficient channel attention for deep convolutional neural networks. *CoRR*, abs/1910.03151, 2019. Disponível em: <http://arxiv.org/abs/1910.03151>.

WANG, S.; ZHANG, H.; ZHANG, X.; SU, Y.; WANG, Z. Voiceprint recognition under cross-scenario conditions using perceptual wavelet packet entropy-guided efficient-channel-attention–res2net–time-delay-neural-network model. *Mathematics*, v. 11, n. 19, 2023. ISSN 2227-7390. Disponível em: <https://www.mdpi.com/2227-7390/11/19/4205>.

WERTZNER, H. F.; SCHREIBER, S.; AMARO, L. Análise da freqüência fundamental, jitter, shimmer e intensidade vocal em crianças com transtorno fonológico. *Revista Brasileira de Otorrinolaringologia*, SciELO Brasil, v. 71, p. 582–588, 2005.

ZHAO, S.; RUDZICZ, F. Classifying phonological categories in imagined and articulated speech. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.: s.n.], 2015. p. 992–996.

ZWICKER, E. Subdivision of the audible frequency range into critical bands (frequenzgruppen). *The Journal of the Acoustical Society of America*, v. 33, n. 2, p. 248–248, 1961.

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de São José do Rio Preto

TERMO DE REPRODUÇÃO XEROGRÁFICA

Autorizo a reprodução xerográfica da presente tese, na íntegra ou em partes, para fins de pesquisa.

São José do Rio Preto, 06 de Agosto de 2022

André Furlan