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

André Furlan

Caracterização de *voice spoofing* para fins de verificação de locutores com base na Transformada *Wavelet* e na análise paraconsistente de características

São José do Rio Preto
2021

André Furlan

Caracterização de *voice spoofing* para fins de verificação de locutores com base na Transformada *Wavelet* e na análise paraconsistente de características

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre 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

2021

F985c

Furlan, André

Caracterização de voice spoofing para fins de verificação de locutores com base na transformada wavelet e na análise paraconsistente de caractersticas / André Furlan. -- São José do Rio Preto, 2021

90 f. : il., tabs.

Dissertação (mestrado) - Universidade Estadual Paulista (Unesp), Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto

Orientador: Rodrigo Capobianco Guido

1. Ciência da computação. 2. Processamento de sinais Técnicas digitais. 3. Sistemas de detecção de intrusão (Medidas de segurança). 4. Intrusion detection systems (Computer security). 5. Reconhecimento de padrões. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

André Furlan

Caracterização de *voice spoofing* para fins de verificação de locutores com base na Transformada *Wavelet* e na análise paraconsistente de características

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre 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. Aleardo Manacero Jr
UNESP – Câmpus de São José do Rio Preto

Prof. Dr. Henrique Dezani
FATEC – Campus de São José do Rio Preto

São José do Rio Preto
25 de Agosto de 2021

À Camila, minha amada esposa e companheira, que me incentiva aos estudos e vibra com nossas conquistas.

Agradecimentos

Ao meu orientador, que com competência e gentileza me ajudou até aqui.

A minha Mãe que, mesmo com dificuldades, conseguiu criar seus filhos para o mundo.

A todos os trabalhadores e trabalhadoras que são os reais motores da nossa sociedade... Que um dia o conhecimento os liberte de suas amarras.

A todos e todas que vieram antes de nós e que desenvolveram a ciência e a tecnologia possibilitando os avanços da humanidade. Sem a dedicação desses e dessas gigantes nada disso seria possível.

A toda comunidade de software livre, pois somos o futuro.

E a todos os criadores e criadoras de conteúdo acadêmico escrito e audiovisual que infelizmente não puderam ser citados... Uma nova geração está se educando melhor graças a vocês.

"O livre arbítrio da humanidade e de seus supostos deuses é uma das ilusões que muitos gostam de acreditar."

-O Autor.

Resumo

Voice spoofing é uma estratégia genérica utilizada para burlar sistemas de autenticação biométrica baseados em identificação por voz. Dentre as diversas possibilidades específicas, os ataques do tipo *playback speech* são os que têm recebido considerável atenção da comunidade científica. Assim, por meio da decomposição dos sinais de voz com *wavelets* e posterior análise das respectivas sub-bandas espectrais BARK e MEL, este trabalho dedica-se a determinar qual a melhor combinação BARK/MEL-wavelet para que se obtenha uma separação máxima entre duas classes: Locuções genuínas e falseadas. Após a apuração da melhor combinação de descritores, realizada por meio da Análise Paraconsistente, os vetores de características oriundos dos sinais de voz são submetidos a ensaios de classificação, variando-se o tamanho do conjunto de treinamento e testes. Utilizando as distâncias Euclidiana e Manhattan, além de Máquinas de Vetores de Suporte (SVM), a acurácia máxima obtida foi de 99,7561% para uma base com 820 sinais, a qual considera-se como um resultado promissor frente àqueles existentes na literatura.

Palavras-chave: Análise paraconsistente. *Voice spoofing*. *Playback speech*. Wavelets.

Abstract

Voice spoofing is a generic strategy designed to circumvent biometric systems based on voice identification. Among a diversity of specific possibilities, playback speech attacks have received considerable attention from the scientific community. Thus, based on speech signals decomposition with wavelets for subsequent BARK and MEL scales spectral analysis, this dissertation aims at determining the best filters and scales to optimally separate between two classes: Genuine and spoofed speech. Once the best combination of descriptors is obtained, based on Paraconsistent Engineering, the feature vectors are subjected to classification, varying the randomly chosen training and test sets in size. Euclidean and Manhattan distances, as well as Support Vector Machine (SVM), were used as classifiers, where the highest value of accuracy was 99.7561% for a dataset with 820 signals. This is a promising result, considering the state-of-the-art in the field.

Keywords: Paraconsistent analysis. Voice spoofing. Playback speech. Wavelets.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

1.1 Considerações Iniciais e Objetivos

A verificação automática de voz, ou biometria de voz, tem atraído cada vez mais a atenção de organizações nacionais e internacionais pois constitui uma alternativa aos sistemas atuais de verificação de usuários, ou ainda, como uma camada adicional de segurança para os sistemas tradicionais.

Em tempos de emergência sanitária nos quais a implementação de sistemas de reconhecimento por imagem de íris tem custos proibitivamente altos e que o reconhecimento por impressões digitais incorre em riscos para a saúde por necessitar de contato com os sensores, o reconhecimento de voz aparece como uma alternativa viável de autenticação.

Assim, com o crescente interesse e uso, consequentemente, crescem também as tentativas de fraude contra tais sistemas. Portanto, é importante que esses consigam diferenciar uma tentativa de autenticação legítima de uma fraudulenta. Dentre as muitas estratégias de *voice spoofing* a *playback speech attack* consiste em reproduzir uma voz genuína usando um dispositivo, enganando assim o mecanismo de autenticação (BAUMANN et al., 2021)(YANG; DAS, 2019)(SARANGI; SAHIDULLAH; SAHA, 2020). Considerando tal contexto, os *voice spoofing attacks* do tipo *playback speech* constituem o tema de estudo deste trabalho.

Particularmente, o objetivo deste trabalho é o de encontrar um conjunto de características que demonstrem ser as mais disjuntas possíveis para fins de separação entre duas classes, isto é, locuções genuínas e falsificadas, com o objetivo de utilizá-las em associação com classificadores do tipo *pattern-matching*, ou seja, não baseados em conhecimento e, assim, obter uma técnica eficiente e veloz para detectar tentativas de burlar os sistemas de verificação de locutores por voz. As caraterísticas examinadas, que foram obtidas com base na transformada *Wavelet-Packet* em tempo discreto (DTWPT) devido as boas resoluções em relação às dimensões de tempo e frequência, foram avaliadas usando a Análise Paraconsistente de acordo com o trabalho GUIDO, R. C. Paraconsistent feature engineering [lecture notes]. *IEEE Signal Processing Magazine*, v. 36, n. 1, p. 154–158, Jan 2019. recentemente publicado. Uma vez avaliadas, os melhores descritores foram isolados para uso em conjunto com duas estruturas de classificação: Uma por distância (Euclidiana ou Manhattan) e outra constituída por uma Máquina de Vetores de Suporte (SVM).

Os resultados dos experimentos descritos neste documento foram apresentados e discutidos detalhadamente, conduzindo a um conjunto de interessantes conclusões, tanto do ponto de vista de processamento digital de sinais quanto de sistemas inteligentes.

Acredita-se, assim, que este trabalho forneça uma interessante contribuição, possibilitando ainda futuras investigações.

1.2 Estrutura do trabalho

No Capítulo 2 é realizada uma revisão dos seguintes conceitos: Amostragem, quantização, caracterização dos processos de produção da voz humana, escalas e energias dos sinais, filtros digitais usando *wavelets* e *wavelet-packets*, engenharia paraconsistente de características, além dos trabalhos correlatos contemplando o estado-da-arte. Em seguida, no Capítulo 3, apresenta-se a abordagem proposta para solucionar o problema considerado nesta investigação. No capítulo 4, são descritos os testes e os resultados obtidos para que, no Capítulo 5, apresentem-se as conclusões. Ao final do documento, encontram-se as referências usadas e um apêndice contendo informações complementares em relação ao estudo.

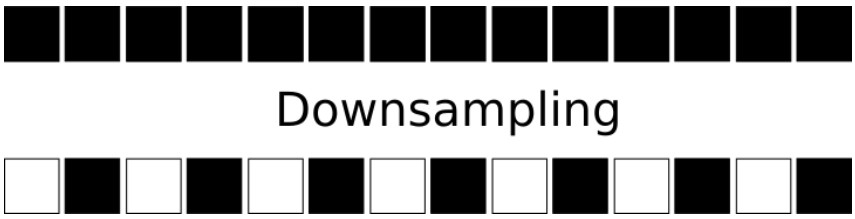
2 Revisão bibliográfica

2.1 Breve revisão dos conceitos utilizados neste trabalho

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

Os sinais de voz digitais, 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 locuções em computadores, os sinais podem sofrer, a depender da necessidade ou possibilidade, sub-amostragens ou *downsamplings* (POLIKAR et al., 1996). Isso implica a 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 filtrados com base em *wavelets*, o processo de sub-amostragem é essencial.

Figura 1 – Sub-amostragem



Fonte: Elaborado pelo autor, 2021.

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 (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 Distâncias Euclidiana e Manhattan

Por definição, as distâncias Euclidiana (D_E) e Manhattan (D_M) entre dois vetores $x[\cdot]$ e $y[\cdot]$ de tamanho M são dadas, respectivamente, por:

$$D_E = \sqrt{\sum_{i=0}^{M-1} (x_i - y_i)^2} \tag{2.1}$$

e

$$D_M = \sqrt{\sum_{i=0}^{M-1} |x_i - y_i|} \quad . \tag{2.2}$$

2.1.4 Escalas e energias dos sinais

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

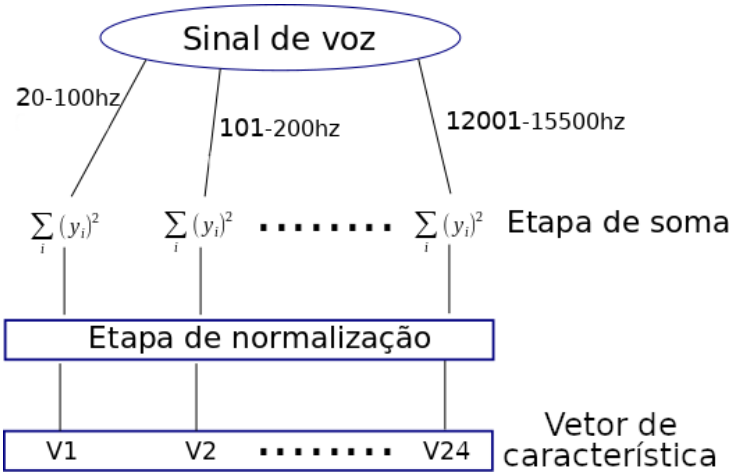
$$E = \sum_{i=0}^{M-1} (s_i)^2 \quad . \tag{2.3}$$

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.

2.1.4.1 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: Elaborado pelo autor, 2021.

2.1.4.2 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, 2006) 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.

2.1.5 Filtros digitais *wavelet*

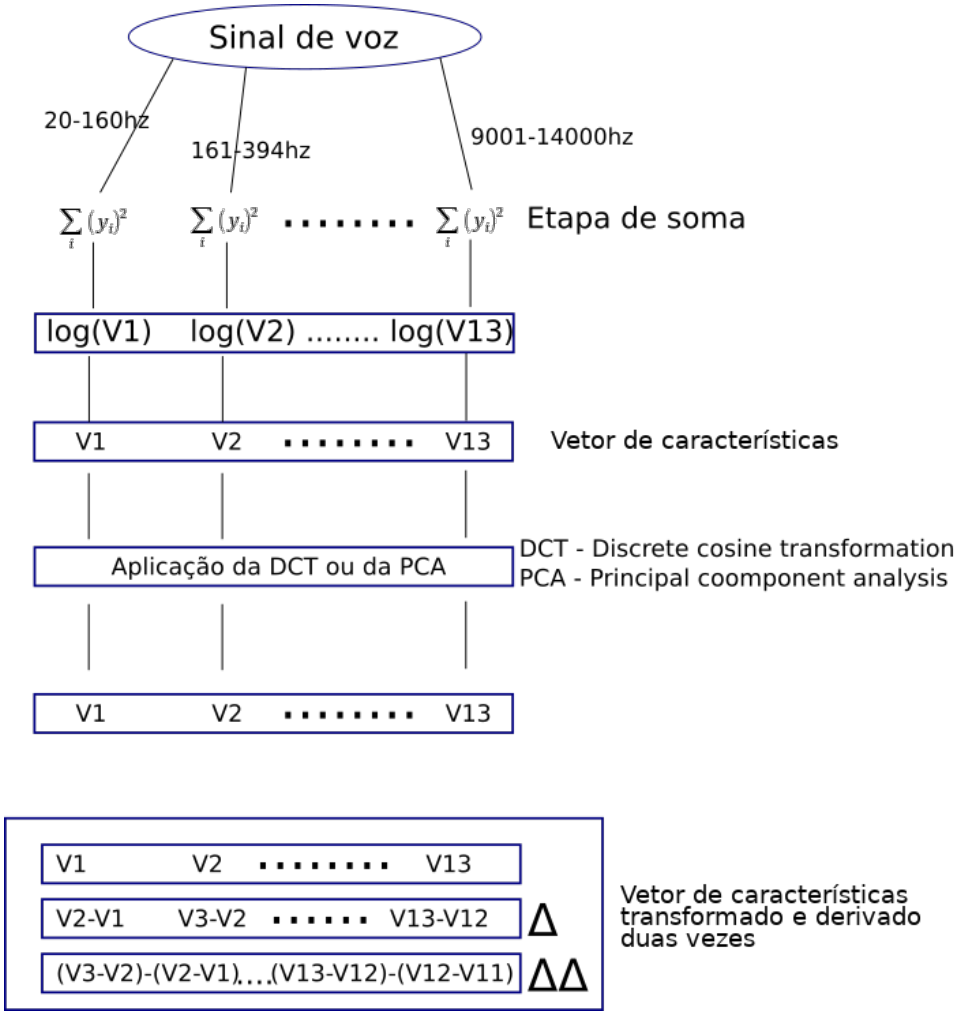
Filtros digitais *wavelet* têm sido utilizados com sucesso para suprir as deficiências de janelamento de sinal apresentadas pelas Transformadas de Fourier e de Fourier de Tempo Reduzido. *Wavelets* contam com variadas funções-filtro e têm tamanho de janela variável, o que permite uma análise multirresolução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

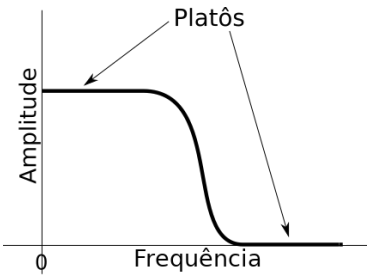
Figura 3 – Cálculo de vetores de características com MEL



Fonte: Elaborado pelo autor, 2021.

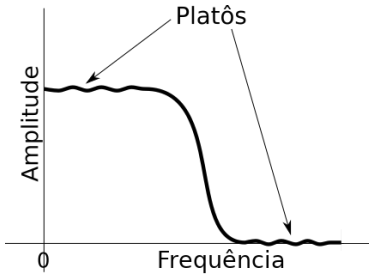
da mesma para filtragem, não são causadas muitas pertubações indesejadas ao sinal, no domínio da frequência. Os filtros *wavelet* de Daubechies (DAUBECHIES, 1992) se destacam nesse quesito por serem *maximamente planos* (*maximally-flat*) (BUTTERWORTH, 1930) (BIANCHI, 2007) nos platôs de resposta em 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: Elaborado pelo autor, 2021.

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



Fonte: Elaborado pelo autor, 2021.

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;
- 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 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, 2021.

2.1.5.1 O algoritmo de Mallat para a Transformada *Wavelet*

Baseando-se no artigo 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., 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:

$$\begin{pmatrix} \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0 \\ \frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}, 0, 0 \\ 0, 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \\ 0, 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \end{pmatrix} \tag{2.4}$$

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:

$$\begin{pmatrix} \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0 \\ \frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}, 0, 0 \\ 0, 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \\ 0, 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} = \begin{pmatrix} \frac{3}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \\ \frac{7}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix} \tag{2.5}$$

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:

$$resultado = \left[\frac{3}{\sqrt{2}}, \frac{7}{\sqrt{2}}, \frac{-1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}\right] \quad . \tag{2.6}$$

Percebe-se que, na transformação descrita nas Equações 2.4, 2.5 e 2.6, 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 2, 3 e 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 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.

2.1.5.2 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

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, 2021.

um nível de detalhes em todo o espectro de frequência, maior do que uma transformação regular.

Os exemplos mostrados nas Tabelas 3 e 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 sub-bandas 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, 2021.

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, 2021.

Para uma visualização mais completa, a Figura 56 pode ser consultada no apêndice deste documento.

2.1.6 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.

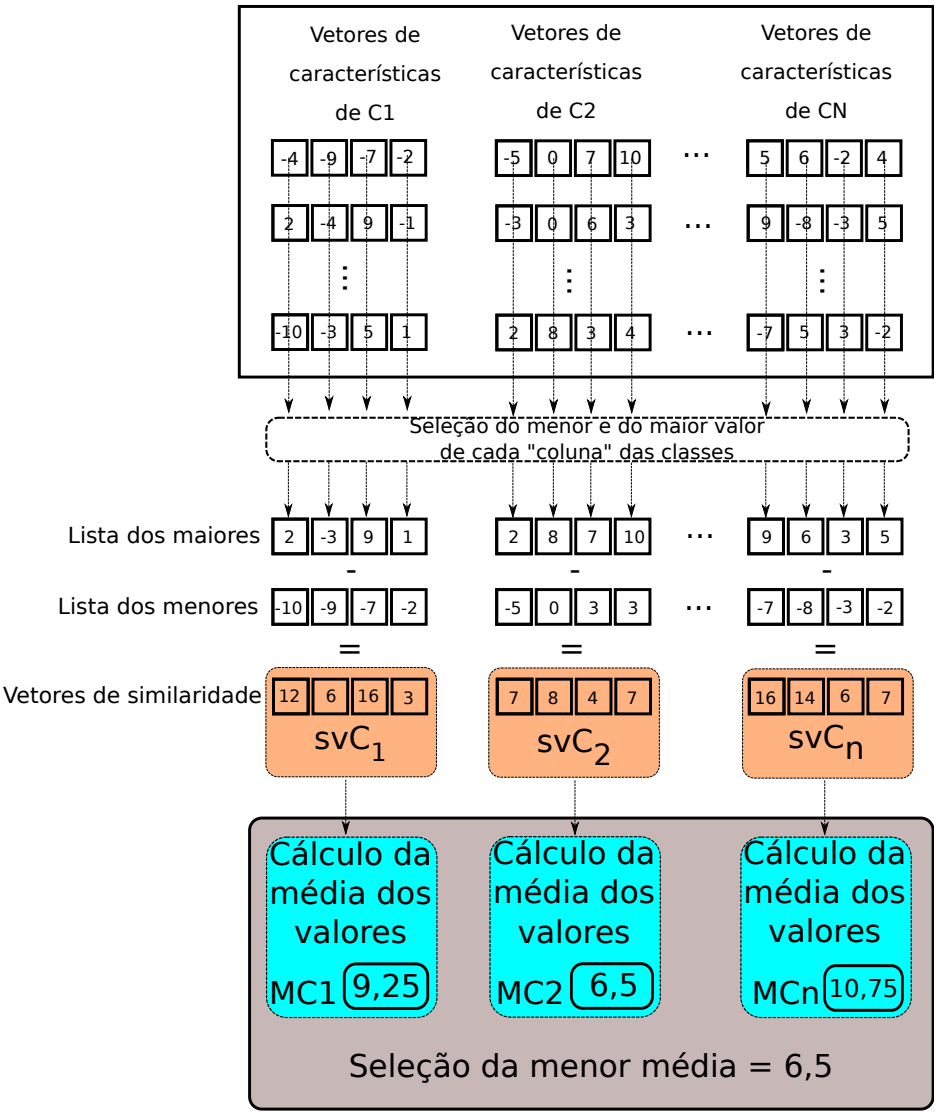
Particularmente, para calcular α e β, é necessária a normalização dos vetores de características de forma que todos os seus componentes estejam no intervalo entre 0 e 1. Em seguida, a obtenção de α se dá selecionando-se os maiores e os menores valores de cada uma das posições de todos os vetores de características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

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.

Figura 6 – Cálculo do coeficiente α.



Fonte: Adaptado de (GUIDO, 2019).

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:

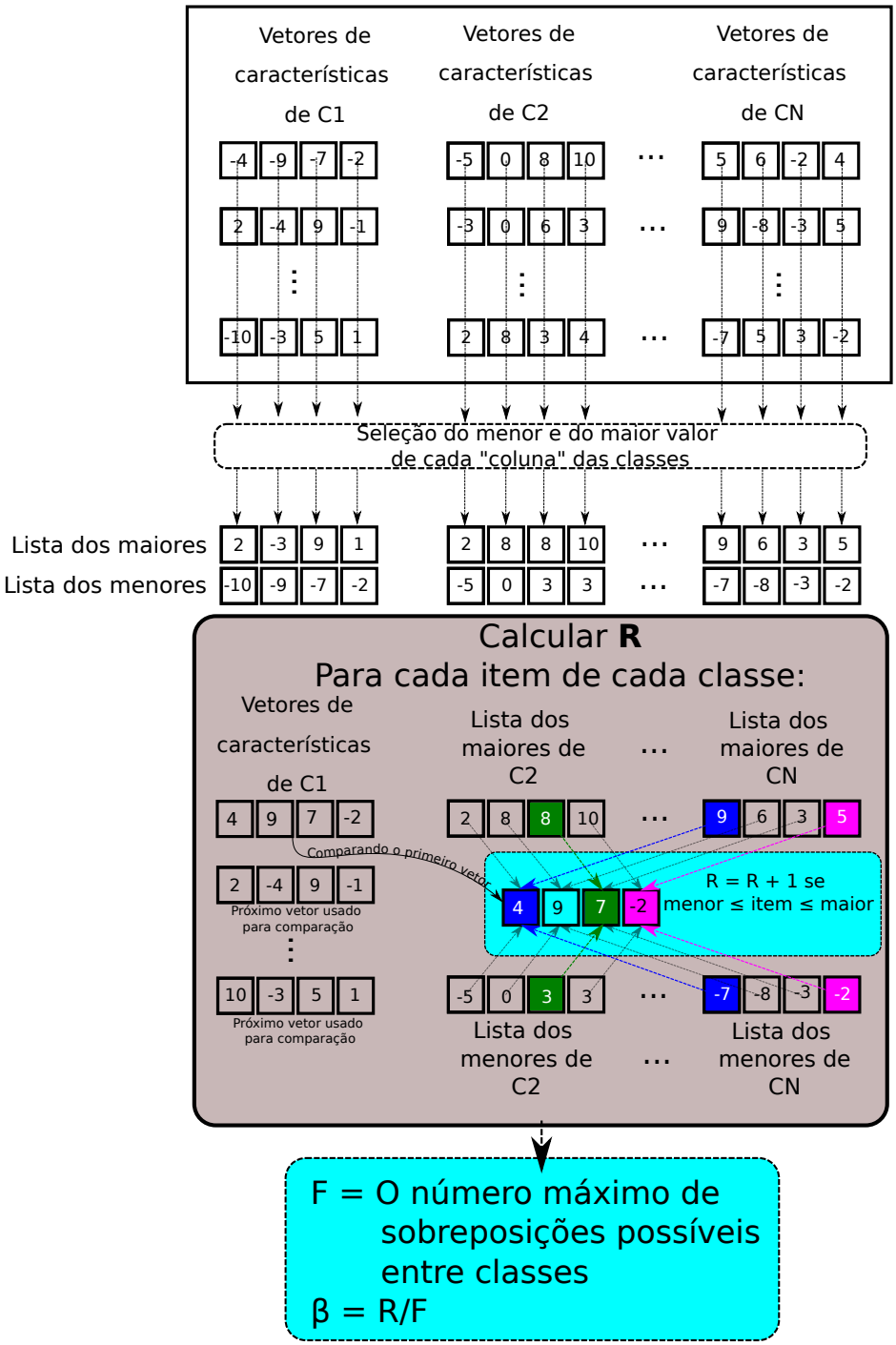
$$F = N.(N-1).X.T \quad . \tag{2.7}$$

Finalmente, β é calculado da seguinte forma:

$$\beta = \frac{R}{F} \quad . \tag{2.8}$$

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).

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.



Fonte: Adaptado de (GUIDO, 2019).

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

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

$$G_1 = \alpha - \beta \quad , \tag{2.9}$$

$$G_2 = \alpha + \beta - 1 \quad , \tag{2.10}$$

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 ($G_2 = 1$). 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:

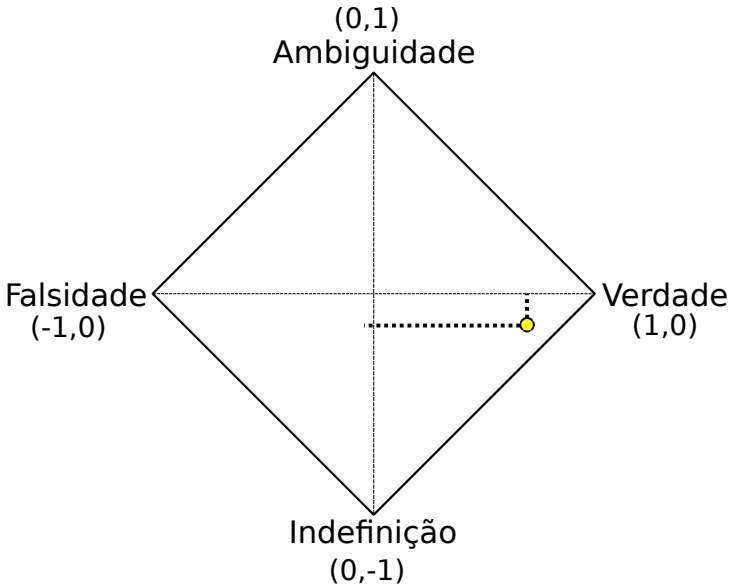
$$D_{-1,0} = \sqrt{(G_1 + 1)^2 + (G_2)^2} \quad , \tag{2.11}$$

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

$$D_{0,-1} = \sqrt{(G_1)^2 + (G_2 + 1)^2} \quad , \tag{2.13}$$

$$D_{0,1} = \sqrt{(G_1)^2 + (G_2 - 1)^2} \quad . \tag{2.14}$$

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)



Fonte: Adaptado de (GUIDO, 2019).

Na prática, ou seja, para fins de classificação, geralmente considera-se a distância em relação ao ponto *"(1,0)* $\rightarrow$ *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 a possibilidade do uso de classificadores mais modestos.

2.2 Estado-da-arte em *Playback Speech Detection*

No artigo REN, Y. et al. Replay attack detection based on distortion by loudspeaker for voice authentication. *Multimedia Tools and Applications*, v. 78, n. 7, p. 8383–8396, Apr 2019. foi apresentado um esquema de diferenciação entre a fala comum e aquela vinda de um dispositivo reprodutor. O foco da análise se deu na distorção causada pelo alto-falante, segundo a energia e outras várias características do espectro do sinal. Uma base com 771 sinais de fala foi criada para cada um dos quatro dispositivos de gravação usados, totalizando 3084 segmentos de voz. Uma SVM foi usada como classificador. De

acordo com os experimentos, a *taxa de verdadeiros positivos* foi de 98,75% e a *taxa de verdadeiros negativos* foi de 98,75%.

Em YAN, D. et al. Detection of hmm synthesized speech by wavelet logarithmic spectrum. *Automatic Control and Computer Sciences*, v. 53, n. 1, p. 72–79, Jan 2019. é mostrado um método para diferenciar a voz de um locutor verdadeiro da voz gerada por sistemas usando sintetizadores baseados no *modelo oculto de Markov* (HMM). SAS 2019(YAMAGISHI, 2019) foi a escolha de base de dados. Este método usa coeficientes de características logarítmicos extraídos de wavelets que são apresentados a um classificador SVM. Os resultados obtidos tiverem, em média, mais de 99% de acurácia.

Usando uma decomposição por espalhamento baseada em *wavelets* e convertendo o resultado em coeficientes cepstrais (SCCs), o artigo LI, K. S. V. S. E. A. H. Front-end for antispoofing countermeasures in speaker verification: Scattering spectral decomposition. *IEEE Journal of Selected Topics in Signal Processing*, v. 11, n. 4, p. 632–643, June 2017. contém a descrição de um vetor de características que é avaliado por Modelos de Misturas Gaussianas (GMM) para fins de *playback speech detection*. SAS 2015 (WU, 2015) foram as bases de dados escolhidas para os testes. Em relação aos resultados, foram usadas a *taxa de falsos verdadeiros* (FAR) que representa a taxa de ocorrências falsas classificadas como verdadeiras e a *taxa de falsos falsos* (FRR) que é a taxa de ocorrências verdadeiras classificadas como falsas. Os pontos em que FAR é igual a FRR foram definidos como pontos de taxa de erros iguais (ERR) e a relação $\frac{FAR}{FRR}$, igual a 0,18 naquele caso, foi usada como parâmetro de avaliação.

Em ALLURI, K. R.; VUPPALA, A. K. Replay spoofing countermeasures using high spectro-temporal resolution features. *International Journal of Speech Technology*, Springer, v. 22, n. 1, p. 271–281, 2019., os autores usam o *zero time windowing* ou janelamento de tempo zero (ZTW) para, em conjunto com a análise cepstral do espectro gerado, fazer a análise dos sinais de voz. Os experimentos foram feitos usando-se a base SAS 2017(KINNUNEN, 2017) com um classificador GMM e a taxa geral de ERR dos experimentos foi de 0,1475.

Em Ye, Y. et al. Detection of replay attack based on normalized constant q cepstral feature. In: . [S.l.: s.n.], 2019. p. 407–411., foi registrado uma diferença entre as propriedades espectrais da voz original e da voz gravada, que pode ser expressa por meio de coeficientes cepstrais. Um GMM foi usado como classificador e a base de dados usada foi a SAS 2017. Quanto aos resultados, foi obtida uma EER geral menor que 0,1.

A proposta do artigo HANILÇI, C. Linear prediction residual features for automatic speaker verification anti-spoofing. *Multimedia Tools and Applications*, v. 77, n. 13, p. 16099–16111, Jul 2018. foi usar sinais residuais de predição linear para, juntamente com coeficientes cepstrais, criar características que foram apresentas a um classificador GMM. Novamente, a base de dados usada foi a SAS 2015 e os resultados em ERR geral foram de 5,249.

Para detecção de *playback speech*, os autores do artigo RAHMENI, R.; AICHA, A. B.; AYED, Y. B. On the contribution of the voice texture for speech spoofing detection. In: . [S.l.]: IEEE, 2019. p. 501–505. importaram, da área de processamento de imagens, o conceito de textura para o processamento de voz. Padrões binários locais (LBP) e seus respectivos histogramas foram usados para a construção do vetor de características que foi avaliado por uma SVM. A base de dados usada para testes foi a SAS 2015 e a taxa máxima de acurácia conseguida foi de 0,7167.

Uma abordagem que combina análise de sinal de fala usando a *Transformada Constante Q* (CQT) com o processamento cepstral foi mostrada no artigo TODISCO, M.; DELGADO, H.; EVANS, N. Constant q cepstral coefficients: A spoofing countermeasure for automatic speaker verification. *Computer Speech & Language*, v. 45, p. 516 – 535, 2017.. Essa técnica resulta no que se chama *Coeficientes Cepstrais de Constante Q*(CQCCs). Segundo o artigo, a vantagem desses coeficientes é a resolução de espectro temporal variável. As bases de dados usadas foram a RedDots (REDDOTS, 2015), SAS 2015 e AVSpoof 2015 (IDIAP RESEARCH INSTITUTE, 2015). Em se tratando de classificadores foram usados dois GMMs, cada um treinado usando os dados genuínos e *spoofing* respectivamente. Os testes realizados para cada uma das bases chegou aos seguintes resultados: SAS 2015 → EER geral de 0.026; AVSpoof 2015 → EER geral de 0; RedDots → EER geral de 0,185.

No artigo PATEL, T.; PATIL, H. Combining evidences from mel cepstral, cochlear filter cepstral and instantaneous frequency features for detection of natural vs. spoofed speech. In: . [S.l.: s.n.], 2015. é usada a *Transformação Auditiva (TU)*, que tem como base a transformada *wavelet*, e a *Cochlear Filter Cepstral Coefficients (CFCC)* que é a junção dos métodos citados mais uma média dos valores em um intervalo de janela definido. Além disso, se define a *estimação da frequência instantânea (IF)* que tem por base a Transformada de Hilbert (JOHANSSON, 1999) e (KSCHISCHANG, 2006). O processo todo tem como objetivo emular mecanismos naturais ocorridos dentro do ouvido e usa, além da *TU*, o cálculo de coeficientes cepstrais e transformada cosseno. Para a composição dos vetores de características foram combinadas as técnicas *MFCC*, *CFCC*, *CFCC+IF*.

A base de dados usada foi a AVSpoof 2015. O classificador utilizado foi um GMM, as classificações chegaram uma EER de 0.083.

O artigo SARANYA, M. S.; PADMANABHAN, R.; MURTHY, H. A. Replay Attack Detection in Speaker Verification Using non-voiced segments and Decision Level Feature Switching. In: IEEE. [S.l.], 2018. (International Conference on Signal Processing and Communications SPCOM), p. 332–336. propõe uma solução para distinguir sinais de voz genuínos daqueles falseados usando reverberação e as partes não vozeadas da fala. Três GMMs foram definidos para a classificação, nesta estratégia os mesmos votam se uma ocorrência é ou não verdadeira, ganhado sempre a classificação que obtiver mais votos. A base de dados utilizada foi a fornecida pelo *"Automatic Speaker Verification and Spoofing Contermeasures Challenge 2017"*(ASVSpoof 2017). O sistema de avaliação de desempenho escolhido, novamente, foi a ERR e esta alcançou um valor de 2,99.

A principal ideia do artigo KAMBLE, M. R.; PATIL, H. A. Novel Variable Length Energy Separation Algorithm using Instantaneous Amplitude Features For Replay Detection. In: Int Speech Commun Assoc. [S.l.], 2018. (Interspeech), p. 646–650. foi a de capturar a amplitude instantânea vinda de flutuações de energia para distinguir entre sinais de voz genuínos daqueles falseados. Segundo os autores, as modulações de amplitude são mais suscetíveis ao ruído inserido no sinal original por uma fonte reprodutora. O estudo usa a base de dados fornecida pelo ASVSpoof 2017 e GMM como classificador. Os resultados apresentados chegaram a uma EER de 0.0019.

No trabalho WICKRAMASINGHE, B. et al. Frequency Domain Linear Prediction Features for Replay Spoofing Attack Detection. In: Int Speech Commun Assoc. [S.l.]: ISCA-INT SPEECH COMMUNICATION ASSOC, 2018. (Interspeech), p. 661–665., foram usadas as diferenças entre bandas de frequências específicas para diferenciar um sinal legítimo de um usado em ataques de falsificação. Particularmente, foi proposta a *predição linear em domínio de frequência*(FDLP) juntamente com GMMs para classificação dos dados presentes na base fornecida pelo ASVspoof 2017. Os resultados apresentados implicam a EER de 0.0803.

No artigo SUTHOKUMAR, G. et al. Modulation dynamic features for the detection of replay attacks. In: . [S.l.: s.n.], 2018. p. 691–695., os autores propuseram duas novas características que visam interpretar as componentes estáticas e dinâmicas do sinal, complementando as características de tempo restrito no espectro, para distinguir entre locuções genuínas e regravadas. São elas a *Modulation Spectral Centroid Frequency* e *Long Term Spectral Average*. O sistema usa como classificador um GMM juntamente com a base

dados fornecida pelo ASVSpoof 2017. Os resultados mostram um valor de EER de 0,0654.

Considerando o envelopamento das amplitudes e frequências instantâneas em cada banda estreita filtrada, os autores do artigo KAMBLE, M. R.; PATIL, H. A. Novel Energy Separation Based Frequency Modulation Features For Spoofed Speech Classification. In: . [S.l.: s.n.], 2017. p. 326–331. discutiram como diferenciar um sinal de voz legítimo de um falseado. A base de dados usada foi a fornecida pelo *"Automatic Speaker Verification and Spoofing Contermeasures Challenge 2015"* (ASVSpoof 2015) e o GMM foi o classificador escolhido. A proposta alcançou a EER de 0,045.

No artigo científico DAS, K. A. et al. Modified Gammatone Frequency Cepstral Coefficients to Improve Spoofing Detection. In: . [S.l.]: IEEE, 2016. p. 50–55., foi proposto o uso do *gammatone frequency cepstral coefficients*(MGFCC). O gammatone é o produto de uma distribuição gamma com um sinal senoidal e é usado na construção de filtros auditivos que, neste caso, são usados para extrair características do sinal de voz. A base de dados usada foi a fornecida pelo ASVspoof 2015 e o classificador usado foi um GMM. Na distinção entre vozes genuínas e regravadas, o EER chegou a 0,02556.

Segundo o artigo AWAIS, A. et al. Speaker recognition using mel frequency cepstral coefficient and locality sensitive hashing. In: IEEE. *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*. [S.l.], 2018. p. 271–276., o *hashing* sensível a *locus* (LSH), que é frequentemente usado como um classificador para problemas relacionados a *big data*, foi combinado com coeficientes MFCCs para distinção entre locuções genuínas e regravadas. No método, os MFCCs foram extraídos dos arquivos de sinal para posterior aplicação do LSH, gerando assim uma tabela *hash*. Esses valores de *hash* foram então comparados, identificando assim o locutor ou locutora. Nos testes realizados houve uma acurácia de 92,66%, a base de dados usada foi a TIMIT 2018 (CONSORTIUM, 2018).

Apesar do uso de *wavelets* em alguns artigos (YAN et al., 2019), (PATEL; PATIL, 2015), (LI, 2017) é interessante notar, pelo menos até o presente momento, que seu uso é escasso em técnicas de prevenção de *voice spoofing*. Em uma das abordagens mais originais que usa as partes não vozeadas do sinal (SARANYA; PADMANABHAN; MURTHY, 2018), seria muito útil o uso das transformadas *wavelet* ou *wavelet packet*. Tanto neste documento como em boa parte das referências utilizadas, é comum o uso de uma escala *MEL* combinada com outras técnicas para construção do método de detecção de *voice spoofing* (HANILÇI, 2018), (PATEL; PATIL, 2015), (AWAIS et al., 2018), (Ye et al., 2019), (SARANYA; PADMANABHAN; MURTHY, 2018). No entanto, em nenhuma dessas

publicações se utilizou a escala *BARK*, que foi, surpreendentemente, a que demonstrou os melhores resultados no caso específico desta dissertação. O uso de coeficientes cepstrais combinados com outras técnicas foi muito comum (ALLURI; VUPPALA, 2019), (LI, 2017), (Ye et al., 2019), (HANILÇI, 2018), (TODISCO; DELGADO; EVANS, 2017), (PATEL; PATIL, 2015), (DAS et al., 2016). Escolhidos os métodos de geração dos vetores de características, foram selecionados classificadores variados, com destaque para o relativamente simples *Gaussian Mixture Models*(GMM), o mais escolhido. Este trabalho também contou com a escolha de dois classificadores modestos: Por distância Euclidiana / Manhattam e Máquina de Vetores de Suporte.

A leitura dos artigos mencionados nesta seção indica ainda que, aparentemente, dentro do contexto de *voice spoofing*, o esforço é para se criar vetores de características cada vez melhores, sendo curioso que em nenhum dos escritos consultados houvesse uma metodologia para comparação de resultados que não fosse a manual. Nesta dissertação, existe uma série de métodos candidatos para geração dos vetores de características baseados em *wavelets* e nas escalas *BARK* e *MEL*. Tais candidaturas foram avaliadas segundo a engenharia paraconsistente de características que selecionou a melhor combinação dentre as opções apresentadas. Em se tratando de resultados, a maioria dos trabalhos, devido ao uso das bases *SAS* e *AVSpoof*, apresentaram seus resultados segundo a *Equal Error Rate (EER)* que é a medida padrão para avaliação. Alguns outros utilizaram somente a acurácia (AWAIS et al., 2018), (RAHMENI; AICHA; AYED, 2019), (YAN et al., 2019), (REN et al., 2019) e medidas calculadas em tabelas de confusão. Neste trabalho, todas as referidas métricas foram consideradas.

3 Abordagem proposta

3.1 A Base de sinais de soz

3.1.1 Coleta dos sinais

Para a realização desta pesquisa, coletou-se uma série de vozes nos arredores do Instituto de Biociências, Letras e Ciências Exatas da UNESP em São José do Rio Preto, no estado de São Paulo. Foram coletadas amostras de 21 indivíduos, das quais 20 foram usadas já que, em um dos casos, não foi possível coletar todos os dados necessários. Tais gravações se constituem de dígitos em um intervalo de 0 a 9 falados tanto em língua Inglesa quanto na Portuguesa. Os locutores foram escolhidos de acordo com seu sexo e idade de forma que a amostra estudada tenha uma abrangência que cubra desde crianças em época pré-escolar até adultos com idades até os 67 anos dos sexos masculino e feminino.

As gravações foram realizadas usando-se um *smartfone* Asus modelo *Ze550kl* rodando o sistema operacional *Android 6.0.1* em ambientes distintos com diferentes níveis de ruído ao fundo, garantindo uma boa variabilidade de interferências corriqueiras, caracterizando casos reais. Foram usados arquivos no formato *wave*, descrito no apêndice B, sem compressão com quantização de 16 bits e taxa de amostragem de 44100Hz, permitindo, segundo o teorema de Nyquist, que sejam registradas frequências de até 22050Hz.

Coletados os sinais, os dígitos pronunciados nos mesmos foram separados um-a-um usando uma ferramenta desenvolvida para esse fim, resultando em um total de 410 sinais de voz de diferentes durações temporais que foram rotulados como “genuínos”. Para cada um deles, foi criado um sinal “espelho”, regravado por um segundo dispositivo, um *notebook* Acer modelo *Travelmate B* com o sistema operacional *Arch GNU/Linux*, caracterizando os 410 sinais rotulados como “falseados”.

3.1.2 Organização da base de sinais

A organização da base de dados se deu por tipo, isto é, genuíno ou falseado, idioma, dígito ditado e interlocutor considerado. Foi criada uma estrutura hierárquica de diretórios de forma a permitir acesso fácil e intuitivo a cada um dos arquivos *wav*, seja por vias automatizadas ou não. Os arquivos falseados residem no diretório “playback”, enquanto que os genuínos se encontram no diretório “live”. Essa organização está ilustrada nas

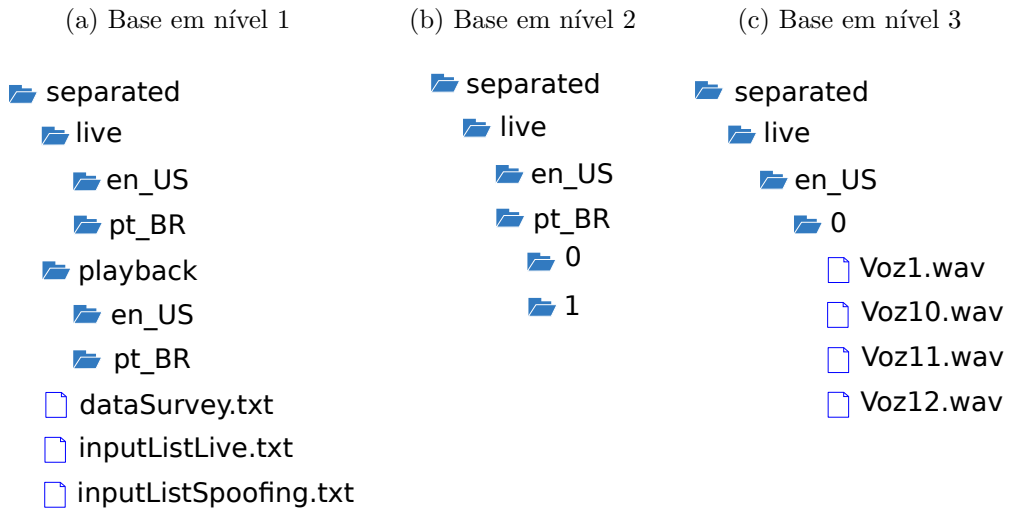
Figuras 9a, 9b e 9c.

Para facilitar a automatização do processamento, foram criados três arquivos de texto:

- ***dataSurvey.txt***: Contêm os dados de idade, sexo e nome do arquivo gerado para cada entrevistado;
- ***inputListLive.txt***: Uma lista de caminhos para todos os arquivos genuínos;
- ***inputListSpoofing.txt***: Apresenta uma listagem dos caminhos para todos os arquivos falseados.

Apenas para ilustrar, o conteúdo do diretório **"separated/live/en_US/0"** se constitui de vários arquivos do tipo *wave*, cada um identificando o locutor ao qual pertence como exibido na Figura 9c.

Figura 9 – Organização da base de dados



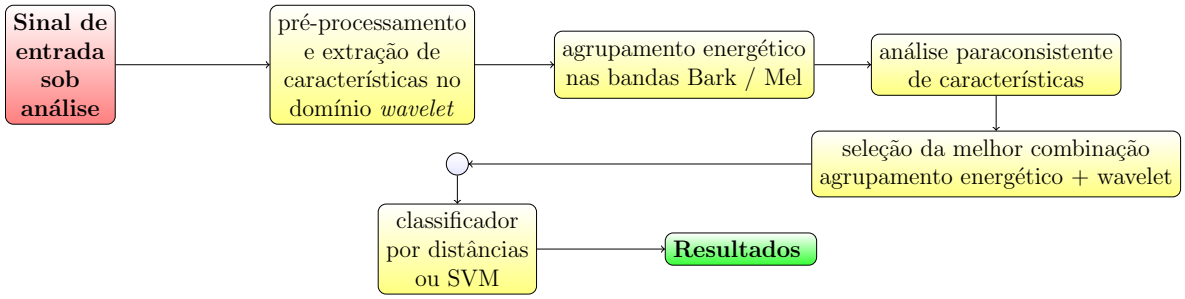
Fonte: Elaborado pelo autor, 2021.

3.2 Estrutura da estratégia proposta

A estratégia proposta para diferenciar sinais de voz genuínos daqueles falseados deu-se conforme ilustrado na Figura 10. Particularmente, a metodologia consiste na obtenção dos dados brutos de todos os 410 + 410 = 820 sinais de voz genuínos e falseados, seguida da conversão de cada um deles para um vetor de características correspondente, conforme explicado adiante. Na sequência, os melhores sub-conjuntos de características foram escolhidos com base na Engenharia Paraconsistente. Prosseguindo, separações aleatórias

entre os vetores, com proporções diversas, foram realizadas para isolar aqueles destinados ao treinamento ou modelagem do classificador utilizado dos que foram destinados aos testes de classificação. Por fim, os testes realizados e seus resultados são mostrados no Capítulo seguinte.

Figura 10 – Estrutura da estratégia proposta



Fonte: Elaborado pelo autor, 2021.

Conforme mencionado nos Capítulos anteriores, os vetores de características desta abordagem foram obtidos com base na Transformada *Wavelet*, convertendo os sinais de voz do domínio do tempo para o domínio tempo-frequência. Particularmente, nos experimentos detalhados adiante, foram testados os seguintes filtros *wavelet*: Haar, Daubechies de suportes 4 até 76, Symmlets com suportes 8, 16 e 32, Coiflets com suportes 6, 12, 18, 24 e 30, Beylkin com suporte 18 e, ainda, Vaidyanathan de suporte 24.

3.3 Procedimentos

De modo a garantir a comparação com outros trabalhos se fez necessário adotar várias formas de representar os resultados correspondentes para cada configuração experimental:

- Tabela de confusão.
- Acurácia e seu respectivo desvio padrão.
- EER (Equal Error Rate).

No exemplo constituído pela tabela de confusão 5 as **linhas** representam as **classes estimadas** e as **colunas** as **classes verdadeiras**, sendo que:

- **TP**: quantidade de itens verdadeiros classificados como tal (*True Positive*).
- **TN**: quantidade de itens falsos classificados como tal (*True Negative*).
- **FN**: quantidade de itens verdadeiros classificados como falsos (*False Negative*).

- **FP**: quantidade de itens falsos classificados como verdadeiros (*False Positive*).

Tabela 5 – Exemplo de matriz de confusão.

	Verdadeiro	**Falso**
Verdadeiro	TP	FP
Falso	FN	TN

Fonte: Elaborado pelo autor, 2021.

A acurácia usa os valores de TP, TN e a quantidade de elementos(N) para ser calculada como mostrado na Equação 3.1.

$$acuracia = \frac{TP + TN}{N} \quad . \tag{3.1}$$

Para se calcular o EER se considera os valores de FP e FN (GHAZALI et al., 2018), a partir desses valores são calculadas as *False Acceptance Rate (**FAR**)* representada na equação 3.2 e *False Rejection Rate (**FRR**)* representada por 3.3 ou **Taxa de falsos positivos** e **Taxa de falsos negativos** respectivamente.

$$FAR = \frac{FP}{TN + FP} \quad . \tag{3.2}$$

$$FRR = \frac{FN}{TP + FN} \quad . \tag{3.3}$$

São calculadas tabelas de confusão por um número suficiente de vezes até que ***FAR* seja igual *FRR***, a cada ciclo os vetores de características são comutados de forma aleatória para que se consiga valores diferentes, neste trabalho alguns casos precisaram de mais que 12000 iterações para que se encontrasse uma configuração em que FAR=FRR.

Em cada iteração os valores de FAR e FRR são guardados em dois vetores, um para cada, então o vetor pertencente a FAR é ordenado de forma crescente e o outro decrescente. Esses pontos desenhados no gráfico com uma linha que divide o plano do mesmo ao meio tal que $x = y$ constituem o que se convencionou chamar de gráfico de *Detection Error Tradeoff (DET)* ou, em tradução livre, **gráfico para balanceamento de erros**.

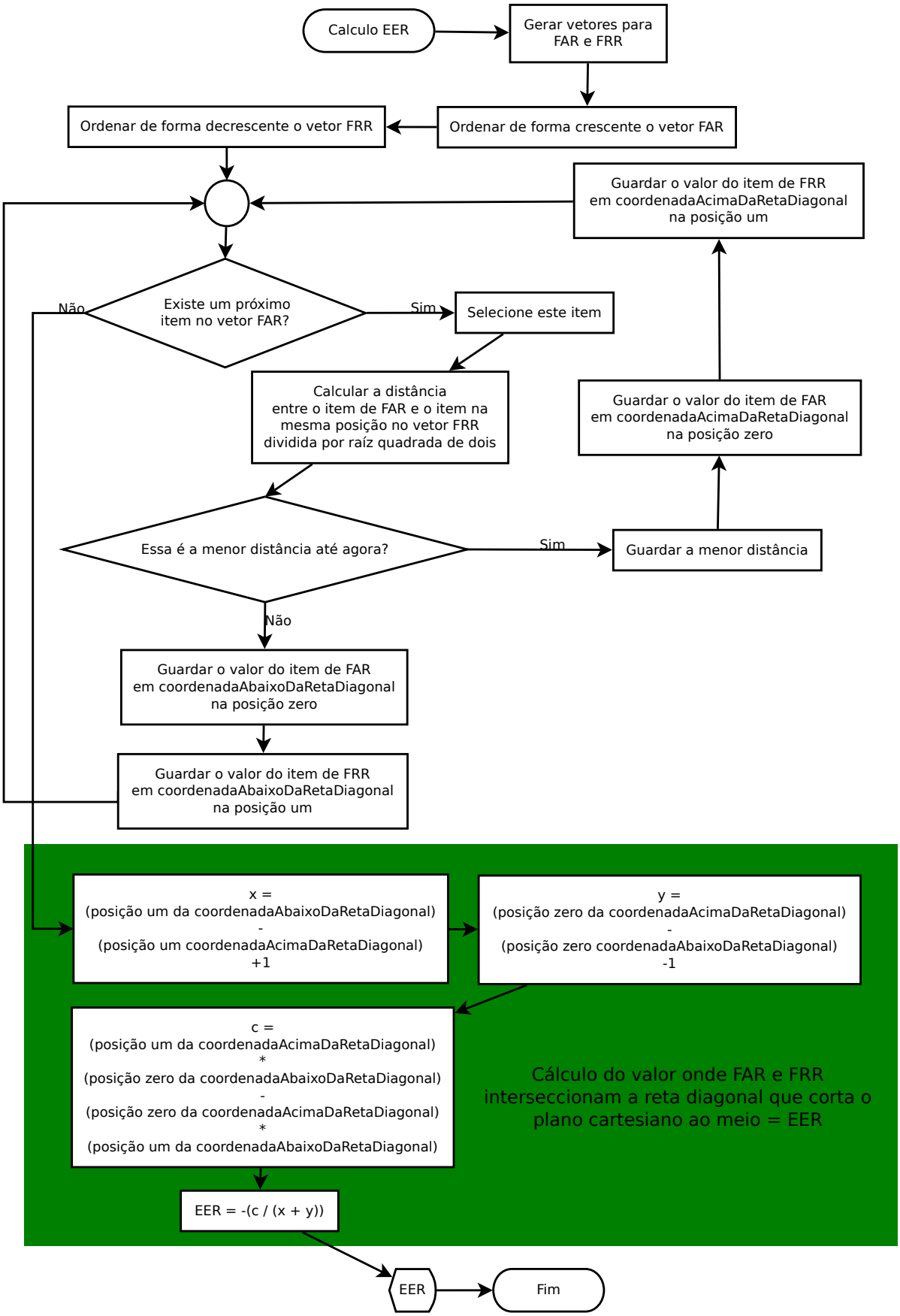
O efetivo valor de **ERR se encontra na intersecção da reta** $x = y$ com a curva definida por FAR e FRR, por isso o cálculo de um valor único como demonstrado no Algoritmo 3.1 e na Figura 11 pois se $x = ERR$ então também $y = ERR$.

Algoritmo 3.1 – Algoritmo para o cálculo do EER

```
double menorDistanciaDaReta = valorMaximo();
double distanciaAtual = -valorMaximo();

double coordenadaAcimaDaRetaDiagonal = { 0, 0 };
double coordenadaAbaixoDaRetaDiagonal = { 0, 0 };

ordenarCrescente(vetorFAR);
ordenarDecrescente(vetorFRR);

for (i = 0; i < vetorFAR.size(); i++) {
  // Calcula a distancia entre a coordenada definida pelos valores de FAR e FRR
  // e a reta definida pela equacao x=y
  distanciaAtual = valorAbsoluto((vetorFAR[i] - vetorFRR[i]) / raizQuadrada(2));

  // Armazena apenas a menor distancia dos pontos acima da reta x=y
  if (distanciaAtual <= menorDistanciaDaReta && vetorFRR[i] >= vetorFAR[i]) {
    menorDistanciaDaReta = distanciaAtual;
    coordenadaAcimaDaRetaDiagonal[0] = vetorFAR[i];
    coordenadaAcimaDaRetaDiagonal[1] = vetorFRR[i];

    pularIteracao();
  }
  // Armazena apenas a menor distancia dos pontos abaixo da reta x=y
  coordenadaAbaixoDaRetaDiagonal[0] = vetorFAR[i];
  coordenadaAbaixoDaRetaDiagonal[1] = vetorFRR[i];

  paraIteracao();
}
// Calcula o ponto de interseccao da curva com a reta x=y
y = coordenadaAcimaDaRetaDiagonal[0] - coordenadaAbaixoDaRetaDiagonal[0] - 1;
x = coordenadaAbaixoDaRetaDiagonal[1] - coordenadaAcimaDaRetaDiagonal[1] + 1;
c = coordenadaAcimaDaRetaDiagonal[1] * coordenadaAbaixoDaRetaDiagonal[0] - coordenadaAcimaDaRetaDiagonal[0] * coordenadaAbaixoDaRetaDiagonal[1];

EER = -(c / (x + y));
```

Fonte: Elaborado pelo autor, 2021.

Figura 11 – Algoritmo para o cálculo do EER



Fonte: Elaborado pelo autor, 2021.

3.3.1 Procedimento 01: Filtros *wavelet*, escalas Bark e Mel

O objetivo deste procedimento é o de verificar, segundo a Engenharia Paraconsistente, qual das combinações entre as escalas BARK ou MEL e as várias *wavelets* consideradas geram os vetores de características mais propícios, ou seja, que atraem o ponto (G_1, G_2) para uma posição mais próxima do vértice $(1, 0)$ do plano paraconsistente, conforme mencionado no Capítulo anterior.

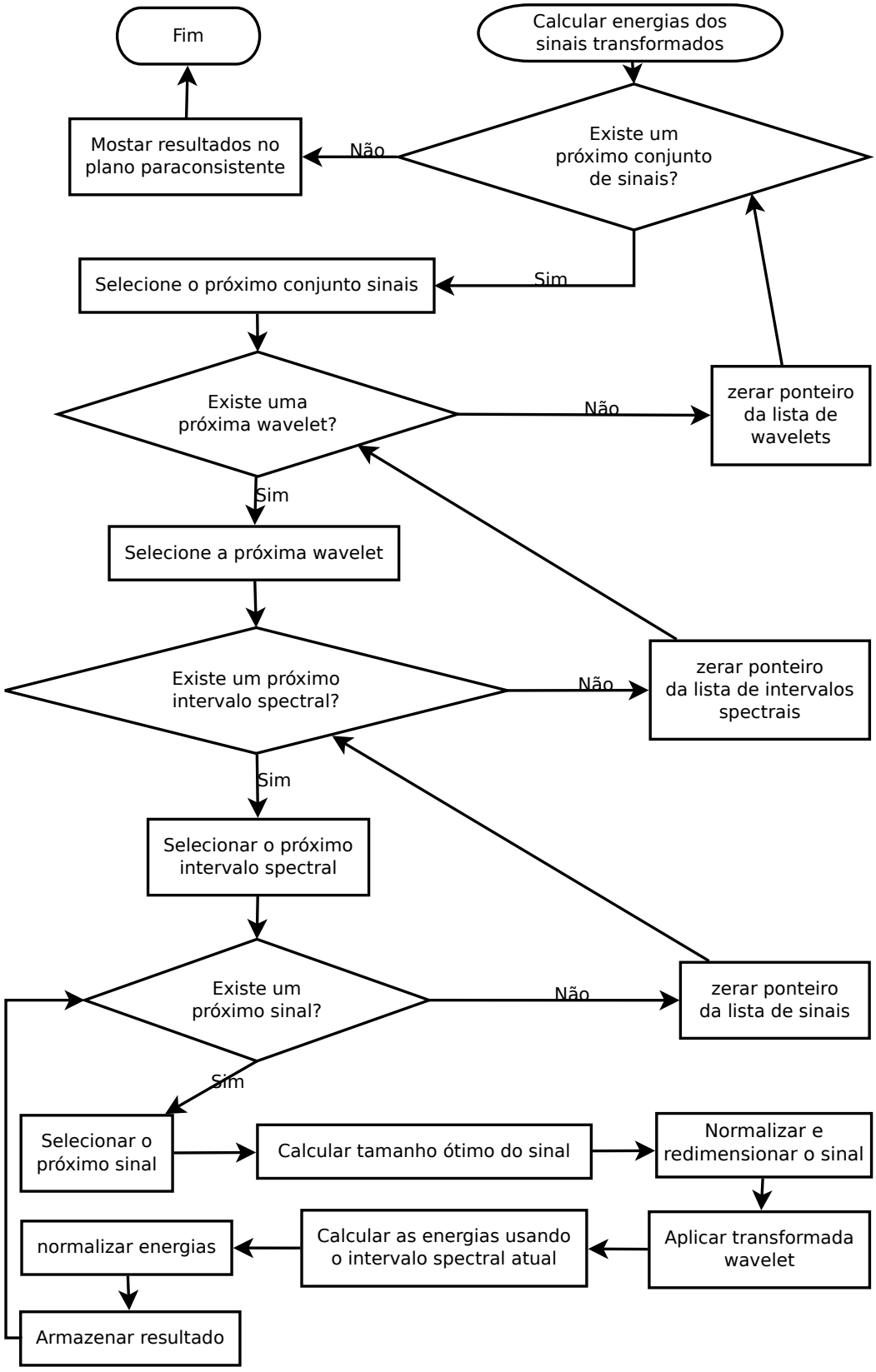
As transformações *wavelet-packet* foram realizadas, com os diversos filtros mencionados, até o máximo nível possível, implicando máxima resolução em frequência para que, após isso, as amostras dos sinais transformados fossem agrupadas visando corresponder aos intervalos espectrais definidos nas escalas BARK e MEL. Naquela escala, os vetores de características foram constituídos de 24 coeficientes, diferentemente, nesta escala, os vetores de características foram formados com 13 coeficientes devido à derivação do sinal ao final do processo de geração. O Algoritmo 3.2 e Figura 12 contém a descrição de tais procedimentos.

Algoritmo 3.2 – Algoritmo que caracteriza o procedimento 01

```
// Carregue para a memoria um dos conjuntos de amostra
for (listaDeAmostras : {listaComVoiceSpoofing, listaSemVoiceSpoofing}) {
  // Selecione o proximo tipo de wavelet
  for (wavelet : wavelets) {
    // Selecione entre BARK ou MEL
    for (barkOuMel : {BARK, MEL}) {
      // Selecione o proximo sinal dentro da amostra
      for (sinal : listaDeAmostras) {
        tamanhoOtimo=calcularTamanhoOtimo(sinal);
        redimensionar(sinal, tamanhoOtimo);
        sinalTransformado=wavelet(sinal, wavelet);
        energias=calcularEnergias(sinalTransformado, barkOuMel);
        energias=normalizar(energias);

        // Armazene os resultados
        resultados[wavelet.nome()][barkOuMel][listaDeAmostras.nome()].
    adicionar(energias);
      }
    }
  }
}
// Posicione os resultados no plano paraconsistente
mostraResultadosNoPlanoParaconsistente(resultados);
```

Fonte: Elaborado pelo autor, 2021.

Figura 12 – Algoritmo que caracteriza o Procedimento 01



Fonte: Elaborado pelo autor, 2021.

Registre-se que, antes da aplicação da transformada *wavelet-packet*, foi necessária uma normalização de valores do sinal entre os intervalos -1 e 1 seguida de um redimensionamento para que houvesse um comprimento correspondente a uma potência de 2, como indicado na equação 3.4. Isso é necessário para que não haja perdas de trechos de voz ao final da transformação, pois a transformada *wavelet* realiza um *downsampling* de ordem 2, ou seja, em cada nível de decomposição o tamanho do vetor do sinal é diminuído pela metade. Caso haja um comprimento diferente do citado, em alguma parte do processo a divisão não será inteira fazendo com que algumas amostras dos sinais sejam perdidas.

Para ajustar o tamanho do sinal de voz sob análise, foi usada a Equação 3.4, na qual ***proxInt*** é uma função que retorna o próximo número inteiro do argumento real. Por exemplo, $proxInt(1,5) = 2$.

$$tamanhoOtimo = 2^{proxInt(\log_2 tamanho)} \tag{3.4}$$

Após o redimensionamento do tamanho do sinal, o nível máximo de transformações é dado pela Equação 3.5.

$$maxTrans = \log_2(tamanhoOtimo) \qquad . \tag{3.5}$$

3.3.2 Procedimento 02 - Classificações baseadas em distâncias

O objetivo deste procedimento é verificar, considerando as melhores combinações descobertas pelo procedimento anterior, a acurácia de classificadores *pattern-matching* por distâncias Euclidiana e Manhattan. Nesta fase, os vetores de características gerados pelo Procedimento 01 são fornecidos aos classificadores para as mensurações devidas.

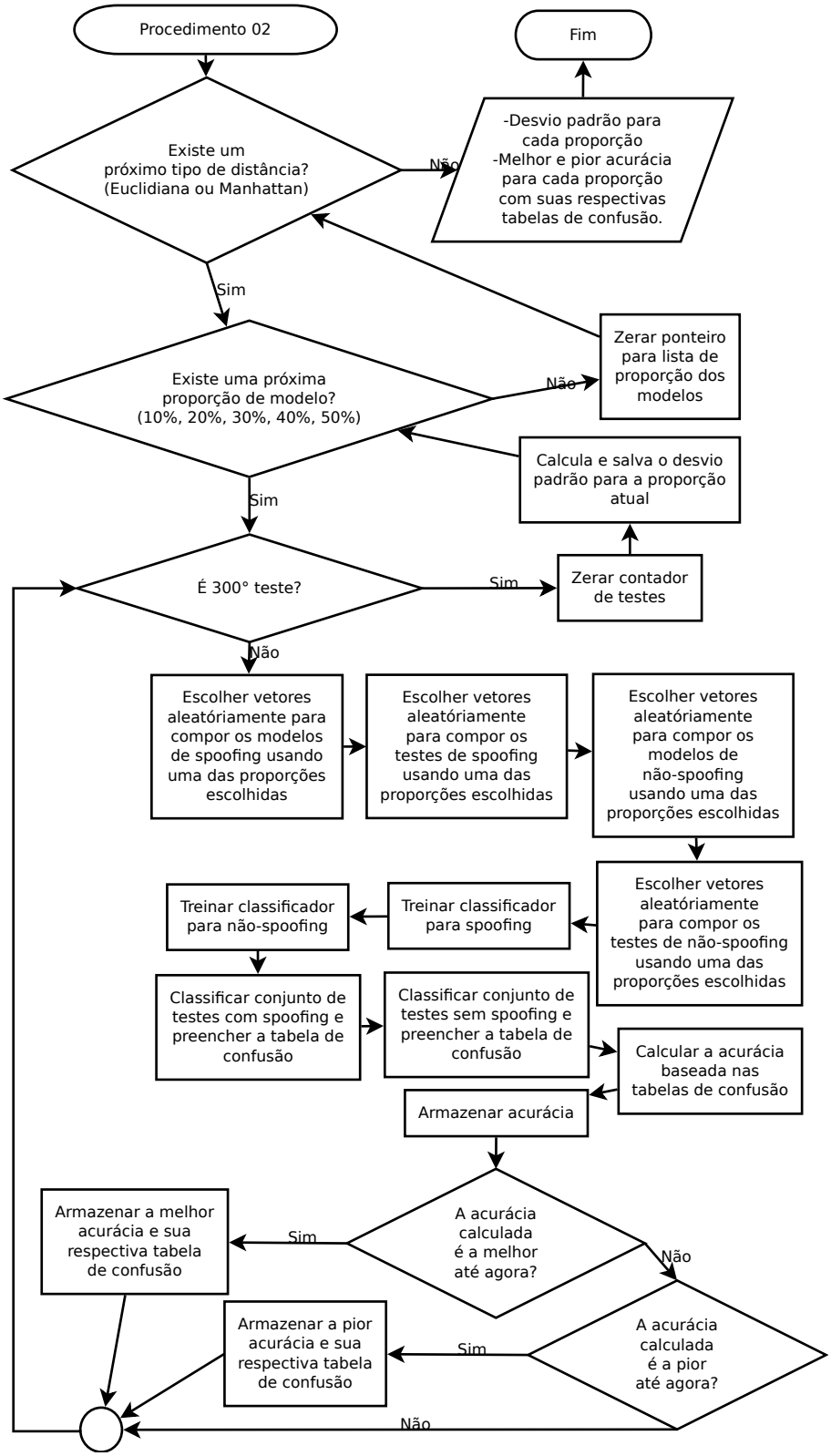
Objetivando avaliar o comportamento dos classificadores com proporções múltiplas de 10% da base de dados reservadas para modelagem até o limite de 50%, e o restante para testes, definiu-se que, para cada proporção, o sorteio aleatório para escolha dos vetores de características e o treinamento dos classificadores seria executado $n = t \cdot \frac{t+1}{2}$ vezes, sendo t o número máximo de testes que podem ser realizados com uma certa porcentagem dos vetores, em outras palavras, **a cada ciclo de treinamento e classificação os vetores nas amostras falseadas e genuínas foram sorteados aleatoriamente de acordo com a porcentagem vigente.**

Para cada porcentagem foram coletadas as melhores e as piores acurácias assim como suas respectivas matrizes de confusão, e assim, calculadas suas *EERs* conforme consta no Capítulo seguinte. No Algoritmo 3.3 e na Figura 13, os passos estão detalhados.

Essencialmente, este Procedimento 02 consiste em mensurar as distâncias entre cada vetor de características isolado para testes em relação à cada um dos vetores de

características isolados para a modelagem, selecionando-se a menor delas. Em seguida, classifica-se o vetor de características que pertence aos testes e está sob análise, como pertencente a uma das classes dos vetores de modelagem selecionados.

Figura 13 – Algoritmo que caracteriza o Procedimento 02



Fonte: Elaborado pelo autor, 2021.

Algoritmo 3.3 – Algoritmo que caracteriza o procedimento 02

```
tamanhosDoModelo={0.1, 0,2, 0,4, 0,5};
modeloDeReferenciaGenuino={};
modeloDeReferenciaFalseado={};
testesGenuino={};
testesFalseado={};

listaDeAcuracias={};

for (distancia : {Euclidiana, Manhattan}) {
  for(porcentagem : tamanhosDoModelo){
    for(teste = 0; teste < 300; teste++){
      // Escolhe aleatoriamente os sinais para o modelo com falseamento
      // e os grava em 'modeloDeReferenciaFalseado' o restante vai
      // para 'testesFalseado'
      escolherAleatoriamente(listaComFalseamento, porcentagem,
   modeloDeReferenciaFalseado, testesFalseado);

      // Escolhe aleatoriamente os sinais para o modelo sem spoofing
      // e os grava em 'modeloDeReferenciaGenuino' o restante vai
      // para 'testesGenuino'
      escolherAleatoriamente(listaGenuino, porcentagem,
   modeloDeReferenciaGenuino, testesGenuino);
      treinarClassificador("falseado", modeloDeReferenciaFalseado);
      treinarClassificador("genuino", modeloDeReferenciaGenuino);

      // Classifica os testes e preenche a tabela de confusao
      for(sinal : testesFalseado){
        preencherTabelaDeConfusao(sinal, "falseado");
      }

      // Classifica os testes e preenche a tabela de confusao
      for(sinal : testesGenuino){
        preencherTabelaDeConfusao(sinal, "genuino");
      }

      acuracia=calculaAcuracia();

      // Guarda as acuracias para cada porcentegem
      listaDeAcuracias[porcentagem].adicionar(acuracia);

      // Salva a melhor acuracia e matriz de confusao
      if(ehAMelhorAcuracia(acuracia)){
        salvaAcuraciaEMatrizDeConfusao();
      }

      // Salva a pior acuracia e matriz de confusao
```

```
45        if(ehAPiorAcuracia(acuracia)){
46          salvaAcuraciaEMatrizDeConfusao();
47        }
48      }
49
50      // Calcula e salva o desvio padrao para cada porcentagem
51      calculaESalvaDesvioPadrao(listaDeAcuracias[porcentagem]);
52    }
53  }
```

Fonte: Elaborado pelo autor, 2021.

3.3.3 Procedimento 03 - classificações baseadas na SVM

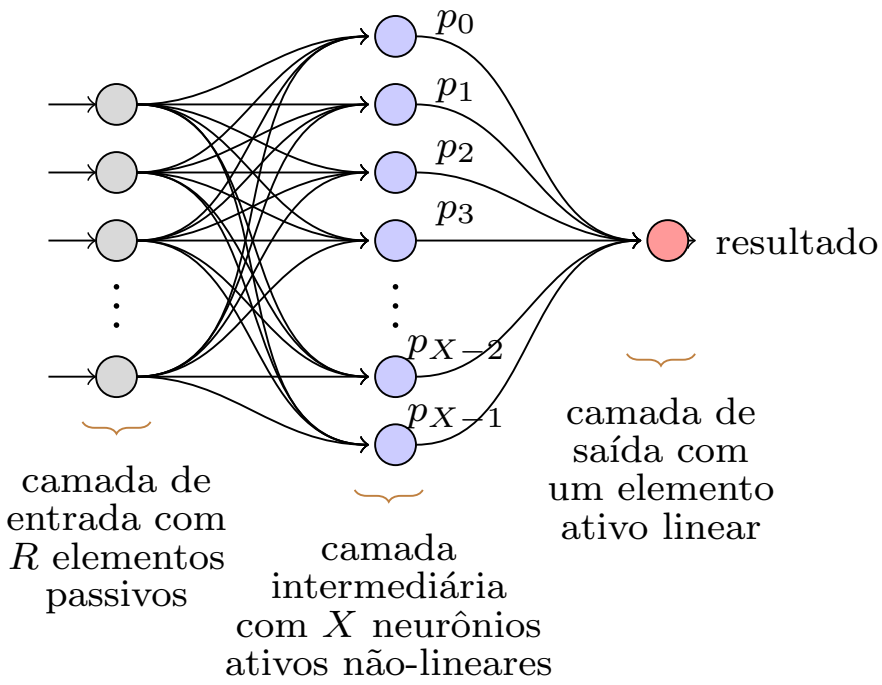
Considerando as melhores combinações descobertas durante o Procedimento 01, esta etapa visa medir a acurácia de uma SVM na separação das classes. O referido classificador foi escolhido, pois, estudos anteriores comprovam a sua eficácia para classificação binária (BENNETT; CAMPBELL, 2000).

Particularmente, a estrutura da SVM utilizada foi definida da seguinte forma e de acordo com a Figura 14:

- três camadas, sendo a primeira de entrada, com elementos passivos, a segunda com elementos ativos não-lineares de núcleos Gaussianos e a terceira, isto é, a de saída, com um elemento ativo linear;
- inexistem pesos entre a camada de entrada e a camada intermediária, implicando que a saída de cada elemento da camada de entrada conecta-se com todas as entradas de cada elemento da camada intermediária, mantendo incólumes os valores propagados;
- a saída de cada elemento da camada intermediária conecta-se com o único elemento da camada de saída por meio dos pesos $p_0, p_1, ...p_{X-1}$;
- o valor de saída consiste na combinação linear dos pesos com os valores recebidos como entrada da camada de saída, isto é, os valores de saída da camada intermediária.

Foram utilizados, na camada de entrada, um número de elementos igual a dimensão do vetor de características sendo considerado. Na camada intermediária, o número de elementos ativos não-lineares foi igual ao número de casos de treinamento, visando facilitar o procedimento que, em tal caso, implica na solução direta de um sistema linear quadrado, isto é, possível e determinado (POOLE, 2014).

Figura 14 – Estrutura da SVM para o procedimento 03 com R neurônios na camada de entrada, sendo R a dimensão dos vetores de características, e X neurônios na camada intermediária, sendo X o número de casos de treinamento



Fonte: Elaborado pelo autor, 2021.

O treinamento da SVM consiste em, numa primeira etapa não-supervisionada, ajustar os parâmetros das funções Gaussianas da camada intermediária. Posteriormente, com base no sistema linear mencionado, os pesos foram encontrados com base em uma abordagem supervisionada, utilizando-se os rótulos -1 e 1 para os sinais falseados e genuínos, respectivamente.

Todos os arranjos para a seleção dos vetores de treinamento e testes, assim como demais detalhes, são idênticos àqueles do procedimento 02 e encontram-se listados no Algoritmo 3.4 e na Figura 15.

Algoritmo 3.4 – Algoritmo que caracteriza o procedimento 03

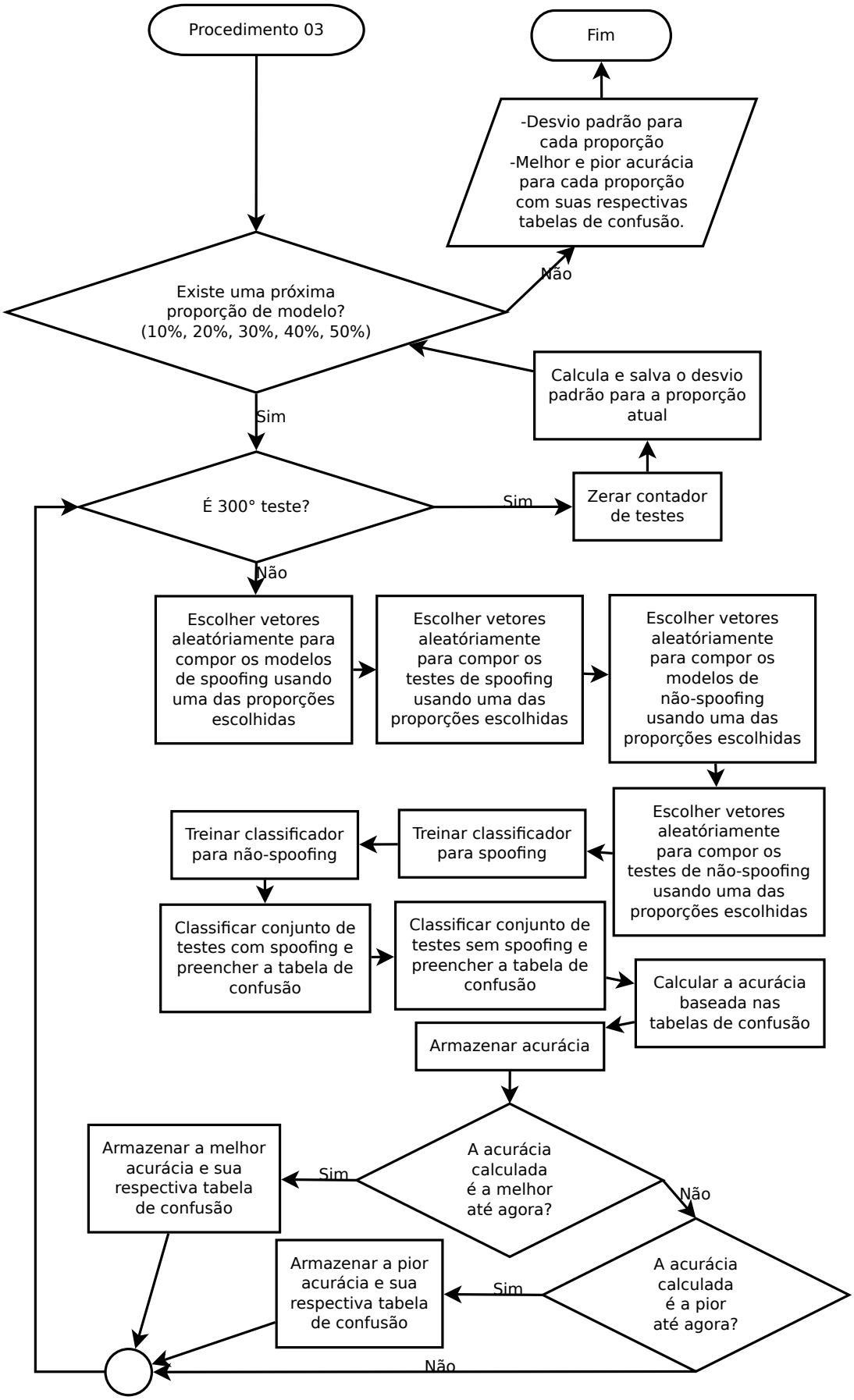
```
tamanhosDoModelo={0.1, 0,2, 0,4, 0,5};
modeloDeReferenciaGenuino={};
modeloDeReferenciaFalseado={};
testesGenuino={};
testesFalseado={};
listaDeAcuracias={};

for(porcentagem : tamanhosDoModelo){
  for(teste = 0; teste < 300; teste++){
    // Escolhe aleatoriamente os sinais para o modelo com falseamento
```

```
11      // e os grava em 'modeloDeReferenciaFalseado' o restante vai
12      // para 'testesFalseado'
13      escolherAleatoriamente(listaComFalseamento, porcentagem,
        modeloDeReferenciaFalseado, testesFalseado);
14
15      // Escolhe aleatoriamente os sinais para o modelo sem spoofing
16      // e os grava em 'modeloDeReferenciaGenuino' o restante vai
17      // para 'testesGenuino'
18      escolherAleatoriamente(listaGenuino, porcentagem,
        modeloDeReferenciaGenuino, testesGenuino);
19      treinarClassificador("falseado", modeloDeReferenciaFalseado);
20      treinarClassificador("genuino", modeloDeReferenciaGenuino);
21
22      // Classifica os testes e preenche a tabela de confusao
23      for(sinal : testesFalseado){
24        preencherTabelaDeConfusao(sinal, "falseado");
25      }
26
27      // Classifica os testes e preenche a tabela de confusao
28      for(sinal : testesGenuino){
29        preencherTabelaDeConfusao(sinal, "genuino");
30      }
31
32      acuracia=calculaAcuracia();
33
34      // Guarda as acuracias para cada porcentegem
35      listaDeAcuracias[porcentagem].adicionar(acuracia);
36
37      // Salva a melhor acuracia e matriz de confusao
38      if(ehAMelhorAcuracia(acuracia)){
39        salvaAcuraciaEMatrizDeConfusao();
40      }
41
42      // Salva a pior acuracia e matriz de confusao
43      if(ehAPiorAcuracia(acuracia)){
44        salvaAcuraciaEMatrizDeConfusao();
45      }
46    }
47
48    // Calcula e salva o desvio padrao para cada porcentagem
49    calculaESalvaDesvioPadrao(listaDeAcuracias[porcentagem]);
50  }
```

Fonte: Elaborado pelo autor, 2021.

Os testes e resultados dos experimentos descritos neste Capítulo encontram-se no Capítulo seguinte.

Figura 15 – Algoritmo que caracteriza o Procedimento 03



Fonte: Elaborado pelo autor, 2021.

4 Testes e Resultados

4.1 Procedimento 01

Os testes, e correspondentes resultados, para o procedimento 01 descrito no Capítulo anterior encontram-se nesta seção. Conforme explicado naquela oportunidade, este procedimento visa, com base na Engenharia Paraconsistente, encontrar qual filtro *wavelet* proporciona, em conjunto com a escala Bark ou Mel, a menor distância do ponto (G_1, G_2) até o vértice $(1, 0)$ do plano paraconsistente.

Particularmente, na Figura 17, que combina os resultados dos filtros com as escalas, quanto menor o comprimento horizontal das barras na cor azul, melhor a separabilidade entre as classes. Como se pode constatar, das combinações testadas, **Haar + Bark** (destacada em verde) proporcionou a menor distância. Complementarmente, a Tabela 6 contém os valores específicos visualizados graficamente naquela Figura.

Em geral, a combinação ***wavelet* + Bark** apresentou **melhor** viabilidade do que as respectivas combinações ***wavelet* + *Mel***.

Tabela 6 – Combinações Wavelet x Mel/Bark ordenadas pela distância do vértice (1,0) no plano paraconsistente.

Mel/Bark	Wavelet	G1	G2	Distância de (1,0)
BARK	haar	-0,019243544792	0,57719954464	1,1713311735
BARK	daub4	-0,10983008668	0,54352357186	1,2357753414
BARK	coif6	-0,113332654	0,54174864681	1,2381442544
BARK	daub6	-0,16108465521	0,53129339358	1,2768673567
BARK	sym8	-0,18796662628	0,53327321112	1,3021693526
BARK	daub8	-0,18903757867	0,53159250263	1,302459578
BARK	coif12	-0,19165365611	0,52724878291	1,3030846923
BARK	daub10	-0,20698255629	0,52665565509	1,3168800516
BARK	daub12	-0,22505599332	0,52728140505	1,333712063
BARK	coif18	-0,22621513169	0,52459787644	1,3337190413
MEL	haar	-0,059434010813	0,8179495812	1,3384475861
BARK	daub14	-0,23587160151	0,51758368304	1,3398774139

Continua na próxima página

Tabela 6: Continuação da página anterior

Mel/Bark	Wavelet	G1	G2	Distance to (1,0)
BARK	sym16	-0,24234940105	0,51679694041	1,3455523445
BARK	daub16	-0,24443938421	0,5156215914	1,3470319248
BARK	coif24	-0,24813974983	0,51476675423	1,3501250484
BARK	daub18	-0,24992752659	0,51927979048	1,3535029821
BARK	daub20	-0,25364592447	0,51657765276	1,3559058136
BARK	coif30	-0,25626533364	0,51111271514	1,3562591183
BARK	daub22	-0,25775073794	0,50901755474	1,3568477401
BARK	daub26	-0,26331722665	0,50660147253	1,3611081761
BARK	daub24	-0,26427150391	0,50432605706	1,3611492231
BARK	daub30	-0,26763536233	0,50075894661	1,3629596232
BARK	daub28	-0,26678413562	0,50618334406	1,3641714057
BARK	daub36	-0,2702330593	0,49816124964	1,3644253939
BARK	daub32	-0,27077597259	0,49700858026	1,3645105721
BARK	sym32	-0,27143585314	0,49563731759	1,3646264981
BARK	daub34	-0,2715180476	0,49809577354	1,3655978709
BARK	beylkin18	-0,27405171882	0,49393608605	1,3664481839
BARK	daub40	-0,2744005356	0,49399377334	1,3667942688
BARK	daub44	-0,27549645406	0,49381248903	1,3677507005
BARK	daub42	-0,27626665556	0,49233090541	1,3679350484
BARK	vaidyanathan24	-0,27613016521	0,49317877788	1,3681131187
BARK	daub38	-0,27512942921	0,4968217903	1,3684980645
BARK	daub50	-0,27806089762	0,48982528124	1,3687105115
BARK	daub46	-0,27687447151	0,49334910572	1,368868787
BARK	daub48	-0,27826990172	0,49134391942	1,3694498124
BARK	daub52	-0,27911603329	0,48948152769	1,3695729234
BARK	daub54	-0,28028206988	0,49034801142	1,3709716812
BARK	daub58	-0,28103680176	0,48908514946	1,3712255726
BARK	daub56	-0,28227860469	0,48967261482	1,3725952389
BARK	daub60	-0,28311882857	0,48842588687	1,3729361876
BARK	daub62	-0,28390708537	0,48885714228	1,3738263025
BARK	daub66	-0,28490636673	0,48806111295	1,3744773629
BARK	daub68	-0,28556278751	0,4876079442	1,3749302484
BARK	daub70	-0,28567912276	0,48738998293	1,3749617457
BARK	daub64	-0,28561054104	0,48816994676	1,3751743017
BARK	daub72	-0,28764126841	0,48695222753	1,3766417501

Continua na próxima página

Tabela 6: Continuação da página anterior

Mel/Bark	Wavelet	G1	G2	Distance to (1,0)
BARK	daub74	-0,28882490282	0,4868864793	1,3777256889
BARK	daub76	-0,28939287641	0,48581037562	1,3778772481
MEL	daub4	-0,10801859422	0,82124969847	1,3791868156
MEL	coif6	-0,11059583194	0,82244186208	1,3819673362
MEL	daub70	-0,12745373447	0,81068373781	1,3886540412
MEL	daub72	-0,12692198418	0,81143721759	1,3886624919
MEL	daub74	-0,12754946358	0,81103146768	1,3889347841
MEL	daub62	-0,12767614873	0,81223515947	1,3897407854
MEL	daub76	-0,12824736379	0,81144221492	1,3897411925
MEL	daub68	-0,1282171885	0,8116941197	1,3898637949
MEL	daub66	-0,12822330339	0,81168800482	1,3898651875
MEL	daub64	-0,12770998385	0,81242305384	1,3898780616
MEL	daub60	-0,12841076059	0,81216573609	1,3902963812
MEL	daub56	-0,12830585727	0,81271409838	1,3905316657
MEL	daub58	-0,12893995529	0,81208000037	1,3906758607
MEL	daub52	-0,12837940132	0,81308401331	1,3908075661
MEL	daub44	-0,12827378554	0,81407654706	1,3913023969
MEL	daub54	-0,12889592735	0,81345440525	1,391443166
MEL	vaidyanathan24	-0,12863649173	0,81437902934	1,391773521
MEL	sym32	-0,12807478314	0,81538419691	1,3919066438
MEL	daub46	-0,12907131045	0,8141659401	1,3920015094
MEL	daub50	-0,12944966121	0,81378758935	1,3920872019
MEL	daub48	-0,12978693016	0,81367204988	1,3922933284
MEL	daub10	-0,12036944258	0,82663720931	1,3923206397
MEL	daub30	-0,12900229067	0,81534360734	1,3926346866
MEL	beylkin18	-0,12908083821	0,81526505979	1,3926523819
MEL	daub36	-0,12919802269	0,81514787532	1,3926787975
MEL	daub16	-0,12537199513	0,82052600931	1,3927401263
MEL	sym16	-0,12541852278	0,82047948165	1,3927503119
MEL	daub18	-0,12621695189	0,81945932306	1,3927951051
MEL	daub28	-0,12877980133	0,81600955565	1,3928443685
MEL	daub38	-0,12953474309	0,81525461389	1,3930142933
MEL	daub42	-0,12989691591	0,81533590005	1,3933555435
MEL	daub40	-0,13025894958	0,81497386638	1,3934373671
MEL	daub12	-0,1231193796	0,82499591973	1,3935621294

Continua na próxima página

Tabela 6: Continuação da página anterior

Mel/Bark	Wavelet	G1	G2	Distance to (1,0)
MEL	daub20	-0,12761152559	0,81895166732	1,3936246216
MEL	sym8	-0,12012088858	0,82954651719	1,3938501459
MEL	coif24	-0,12692058979	0,82052952108	1,3939937986
MEL	daub32	-0,13021248658	0,81612897684	1,3940755975
MEL	daub34	-0,13014161253	0,81642158037	1,3941894639
MEL	daub6	-0,11917748044	0,83159857277	1,3943150358
MEL	daub14	-0,12507512948	0,82370535833	1,3943760484
MEL	daub26	-0,12974180183	0,81748657955	1,3944894573
MEL	daub22	-0,12903104929	0,81864079106	1,3945909275
MEL	coif30	-0,12884780845	0,81904576139	1,3946803698
MEL	coif18	-0,12424583785	0,82542156792	1,3947220041
MEL	daub24	-0,12975547353	0,81880328478	1,3952728225
MEL	coif12	-0,12158785943	0,83051857073	1,3956075461
MEL	daub8	-0,12305703861	0,83171014543	1,3974973624

Fonte: Elaborado pelo autor, 2021.

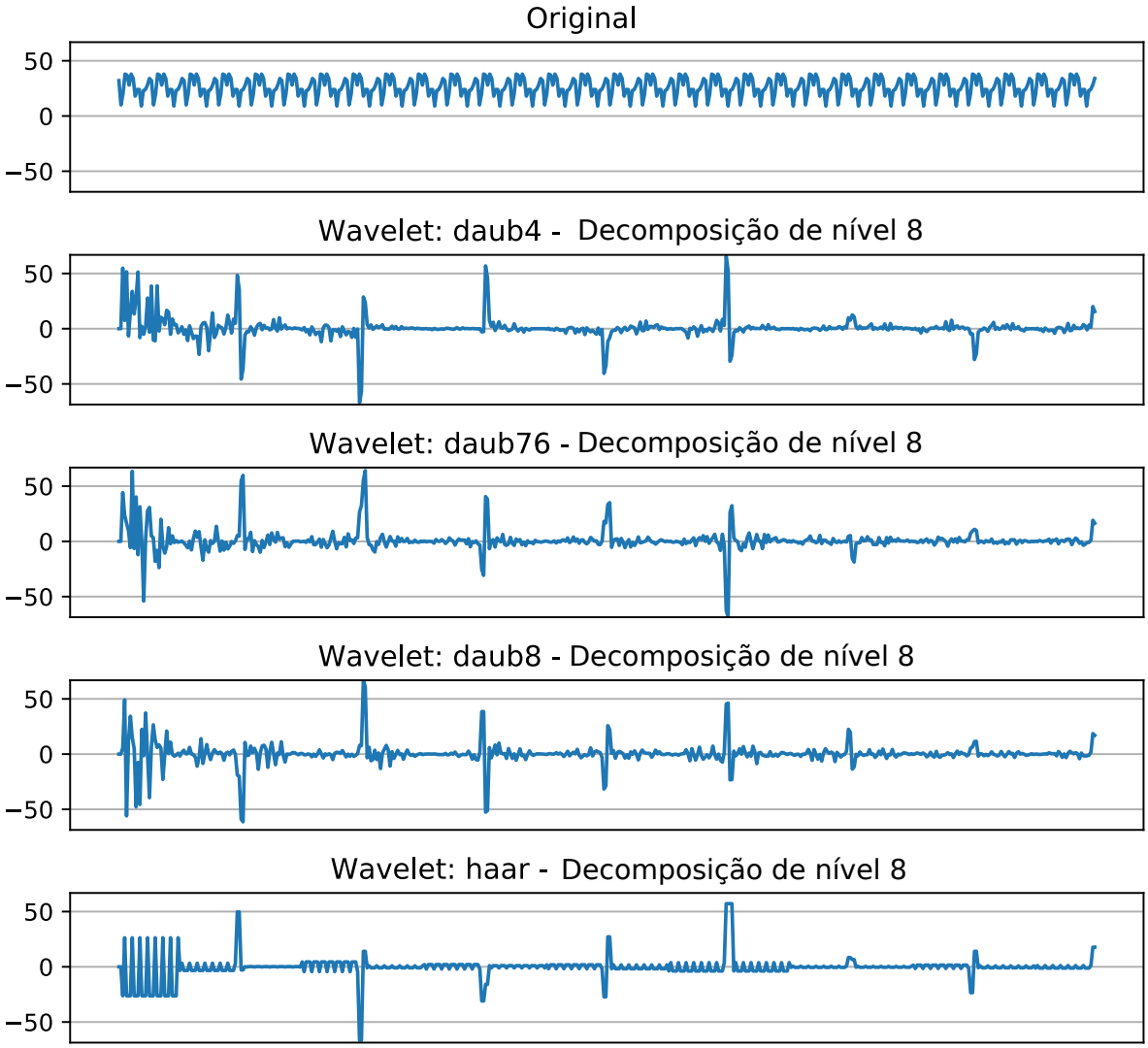
Analisando o comportamento dos melhores e piores resultados encontrados relacionados as *wavelets*, é interessante indagar o motivo pelo qual o filtro de **Haar** tenha proporcionado o melhor resultado. Particularmente interessante é o fato de que os filtros *wavelet-packet* de **Haar** e **Daubechies 76** proporcionaram, respectivamente, os melhores e os piores resultados associados com a escala Bark. Diferentemente, com a escala *Mel*, **Haar** foi o melhor filtro e **Daubechies 8** o pior.

Na Figura 16 é possível ver um sinal periódico decomposto em suas componentes usando o melhor e os piores filtros já citados: Nota-se que com a wavelet **Haar** a decomposição tem menos flutuações, facilitando portanto o treinamento do classificador.

Refletindo acerca das características de resposta em fase e em frequência dos filtros *wavelet*, pode-se constatar o seguinte: Haar apresenta a curva de resposta em frequência mais distante da ideal, pois constitui um filtro de resposta ao impulso finita (FIR) de ordem 1, isto é, com dois coeficientes (WASILEWSKI, 2020). Assim sendo, sub-bandas específicas de frequências estão, sem dúvidas, contaminadas com conteúdo espectral das sub-bandas adjacentes. Adicionalmente, Haar é a única família de filtros *wavelet* que possui resposta em fase perfeitamente linear. Portanto, do ponto de vista dos filtros, o que foi constatado experimentalmente é que uma resposta em frequência não rigorosa associada a uma resposta em fase perfeitamente linear é a melhor alternativa.

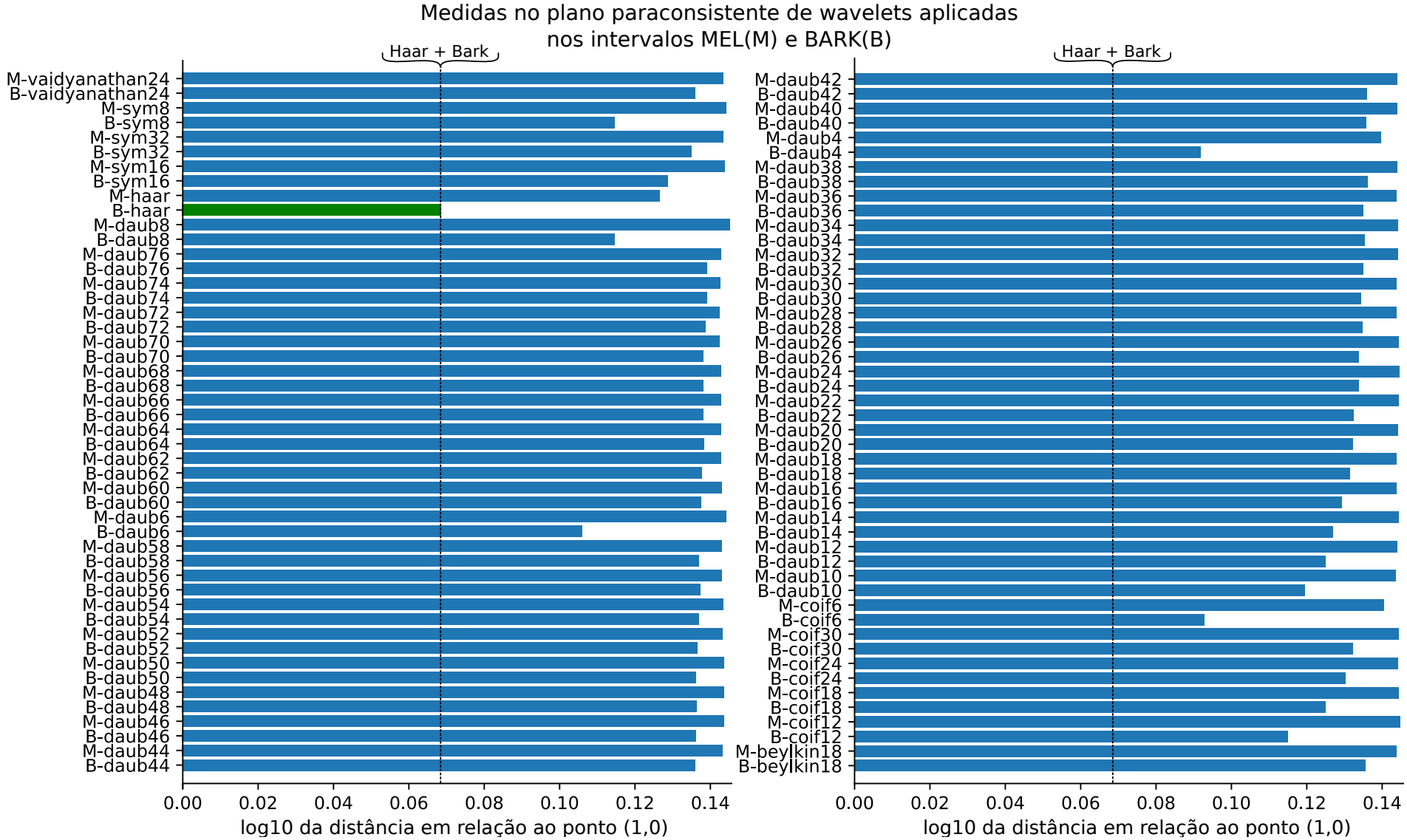
Do ponto de vista das bandas Bark e Mel, aquela é, de acordo com os comentários constantes no Capítulo 2, a que foi definida de modo mais amplo para refletir o comportamento da audição humana para sinais acústicos em geral, diferentemente, esta banda foi otimizada para voz, mas de acordo com os experimentos, não proporcionou os melhores resultados. Assim sendo, fica claro que a **característica ruidosa** dos sinais falseados, contendo, ao contrário das vozes originais, notórios componentes de altas frequências, é melhor detectada com uma escala mais apropriada ao tratamento de áudio em geral e não voz somente.

Figura 16 – Comparação das decomposições de um sinal de tamanho 512 usando os filtros wavelet de melhor e piores desempenhos



Fonte: Elaborado pelo autor, 2021.

Figura 17 – Distâncias de P=(G1, G2) ao ponto (1, 0) no plano paraconsistente. M=Mel; B=Bark; Sym=Symlet; Daub=Daubechies; Coif=Coiflet. Para uma melhor visualização, foi aplicado o log_{10} as distâncias correspondentes.



Fonte: Elaborado pelo autor, 2021.

4.2 Procedimento 02

Considerando que, de acordo com o procedimento 01, o melhor resultado foi a combinação **Haar + Bark**, o objetivo deste procedimento é **constatar a máxima acurácia e o mínimo EER** que se consegue atingir com o uso de um classificador baseado em distâncias Euclidianas e Manhattan, de acordo com os detalhes definidos no Capítulo anterior.

Assim sendo, constatou-se, para os montantes de 10%, 20%, 30%, 40% e 50% da base de sinais reservados para treinamento, com o limite de 300 testes aleatórios em cada caso, os resultados constantes nas Tabelas 7 e 8. Particularmente interessante nelas, é a medida da taxa de erros iguais (EER), que equilibra as taxas de falsos positivos e falsos negativos.

Maiores detalhes para a distância Euclidiana podem ser conseguidos consultando-se as Tabelas 9a, 9b, 10a, 10b, 11a, 11b, 12a, 12b, 13a, 13b e seus respectivos gráficos nas Figuras 18, 20, 22, 24 e 26. Para a distância Manhattan, podem ser consultadas as Tabelas 14a, 14b, 15a, 15b, 16a, 16b, 17a, 17b, 18a, 18b e seus respectivos gráficos nas Figuras 28, 30, 32, 34, 36.

Tabela 7 – Resultados da abordagem de *pattern-matching* com distância Euclidiana.

M	**Acurácia mínima**	**Acurácia máxima**	**Média das acurácias**	**Desvio padrão da acurácia**	**EER**
10%	0,859079	0,9729	0,9159895	0,012382	0,056911
20%	0,900915	0,97561	0,9382625	0,008875	0,04878
30%	0,914634	0,979094	0,946864	0,007918	0,04878
40%	0,920732	0,98374	0,952236	0,007843	0,044715
50%	0,917073	0,987805	0,952439	0,008261	0,039024

Fonte: Elaborado pelo autor, 2021.

Tabela 8 – Resultados da abordagem de *pattern-matching* com distância Manhattan.

M	**Acurácia mínima**	**Acurácia máxima**	**Média das acurácias**	**Desvio padrão da acurácia**	**EER**
10%	0,876694	0,97561	0,926152	0,012179	0,051491
20%	0,900915	0,980183	0,940549	0,008982	0,045732
30%	0,912892	0,986063	0,9494775	0,008087	0,041812
40%	0,928862	0,985772	0,957317	0,007982	0,04065
50%	0,926829	0,990244	0,9585365	0,008347	0,039024

Fonte: Elaborado pelo autor, 2021.

4.2.1 Resultados para escala BARK com *wavelet* Haar usando um classificador por distâncias Euclidiana/Manhattan

Tabela 9 – Matrizes de confusão para distância Euclidiana com modelo a 10%

(a) Melhor matriz de confusão

	genuíno	falseado
genuíno	363	14
falseado	6	355

(b) Pior matriz de confusão

	genuíno	falseado
genuíno	275	10
falseado	94	359

Fonte: Elaborado pelo autor, 2021.

Tabela 10 – Matrizes de confusão para distância Euclidiana com modelo a 20%

(a) Melhor matriz de confusão

	genuíno	falseado
genuíno	323	11
falseado	5	317

(b) Pior matriz de confusão

	genuíno	falseado
genuíno	279	16
falseado	49	312

Fonte: Elaborado pelo autor, 2021.

Tabela 11 – Matrizes de confusão para distância Euclidiana com modelo a 30%

(a) Melhor matriz de confusão

	genuíno	falseado
genuíno	283	8
falseado	4	279

(b) Pior matriz de confusão

	genuíno	falseado
genuíno	258	20
falseado	29	267

Fonte: Elaborado pelo autor, 2021.

Tabela 12 – Matrizes de confusão para distância Euclidiana com modelo a 40%

(a) Melhor matriz de confusão

	genuíno	falseado
genuíno	243	5
falseado	3	241

(b) Pior matriz de confusão

	genuíno	falseado
genuíno	219	12
falseado	27	234

Fonte: Elaborado pelo autor, 2021.

Tabela 13 – Matrizes de confusão para distância Euclidiana com modelo a 50%

(a) Melhor matriz de confusão

	genuíno	falseado
genuíno	204	4
falseado	1	201

(b) Pior matriz de confusão

	genuíno	falseado
genuíno	185	14
falseado	20	191

Fonte: Elaborado pelo autor, 2021.

Tabela 14 – Matrizes de confusão para distância Manhattan com modelo a 10%

(a) Melhor matriz de confusão

	genuíno	falseado
genuíno	365	14
falseado	4	355

(b) Pior matriz de confusão

	genuíno	falseado
genuíno	289	11
falseado	80	358

Fonte: Elaborado pelo autor, 2021.

Tabela 15 – Matrizes de confusão para distância Manhattan com modelo a 20%

(a) Melhor matriz de confusão

	genuíno	falseado
genuíno	321	6
falseado	7	322

(b) Pior matriz de confusão

	genuíno	falseado
genuíno	280	17
falseado	48	311

Fonte: Elaborado pelo autor, 2021.

Tabela 16 – Matrizes de confusão para distância Manhattan com modelo a 30%

(a) Melhor matriz de confusão

	genuíno	falseado
genuíno	281	2
falseado	6	285

(b) Pior matriz de confusão

	genuíno	falseado
genuíno	256	19
falseado	31	268

Fonte: Elaborado pelo autor, 2021.

Tabela 17 – Matrizes de confusão para distância Manhattan com modelo a 40%

(a) Melhor matriz de confusão

	genuíno	falseado
genuíno	244	5
falseado	2	241

(b) Pior matriz de confusão

	genuíno	falseado
genuíno	218	7
falseado	28	239

Fonte: Elaborado pelo autor, 2021.

Tabela 18 – Matrizes de confusão para distância Manhattan com modelo a 50%

(a) Melhor matriz de confusão

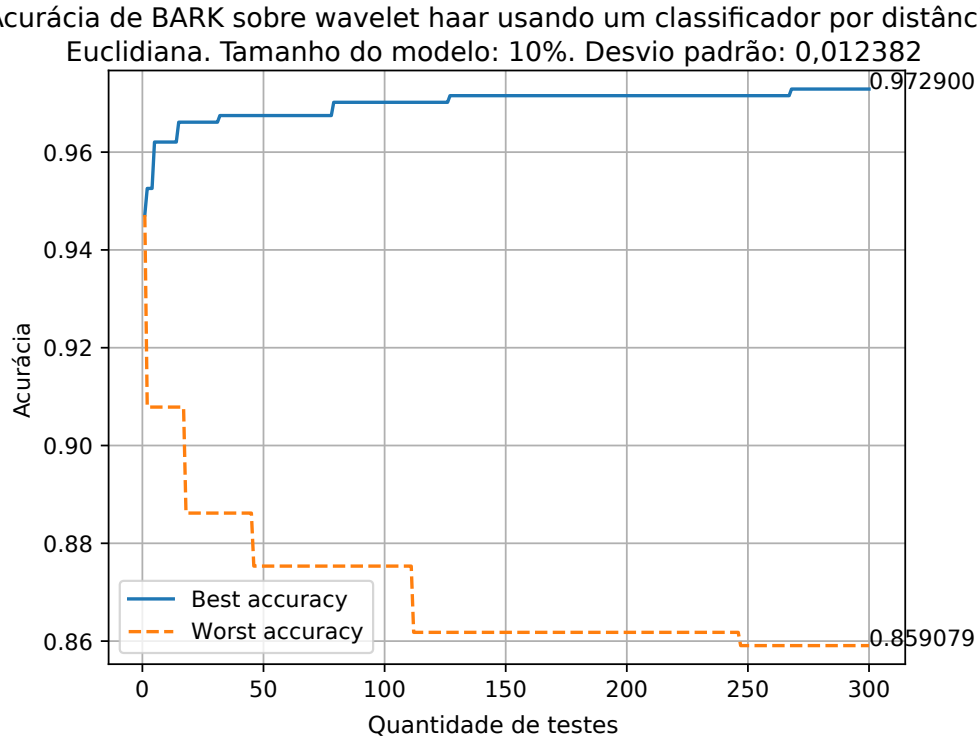
	genuíno	falseado
genuíno	203	2
falseado	2	203

(b) Pior matriz de confusão

	genuíno	falseado
genuíno	188	13
falseado	17	192

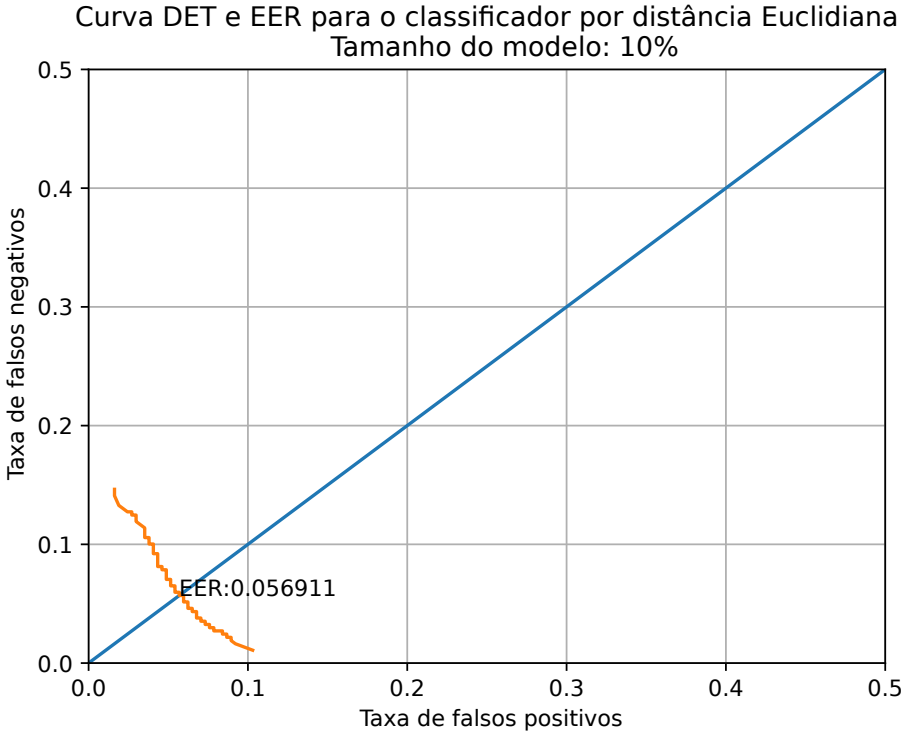
Fonte: Elaborado pelo autor, 2021.

Figura 18 – Acurácia x quantidade de testes - Distância Euclidiana, modelo a 10%



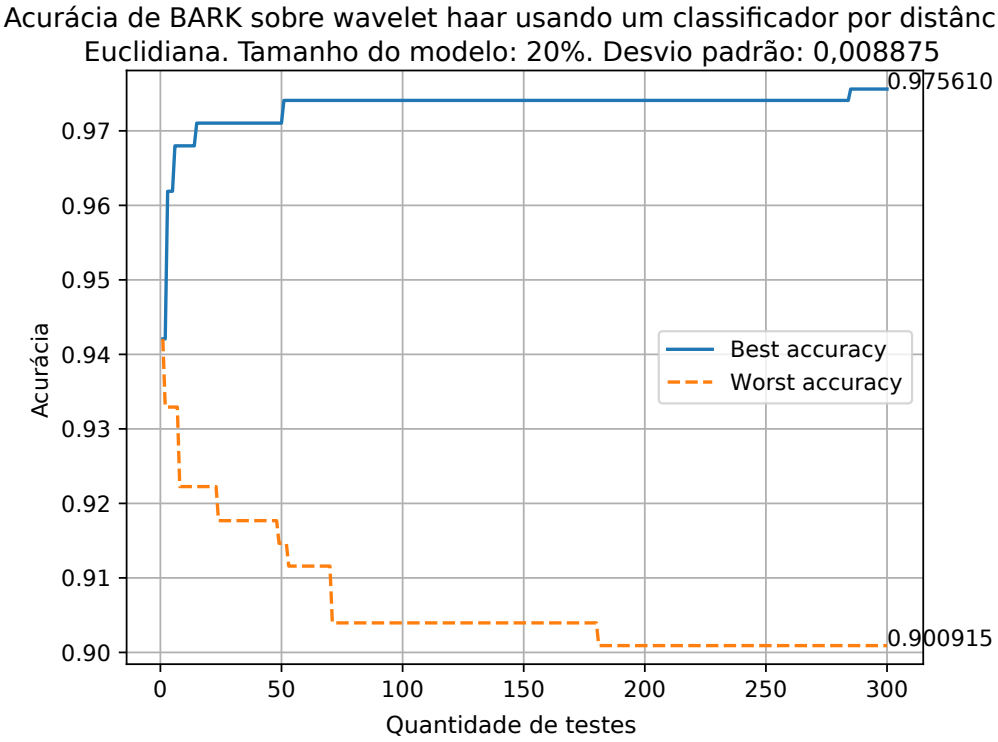
Fonte: Elaborado pelo autor, 2021.

Figura 19 – Curva DET dos resultados de distância Euclidiana, modelo a 10%



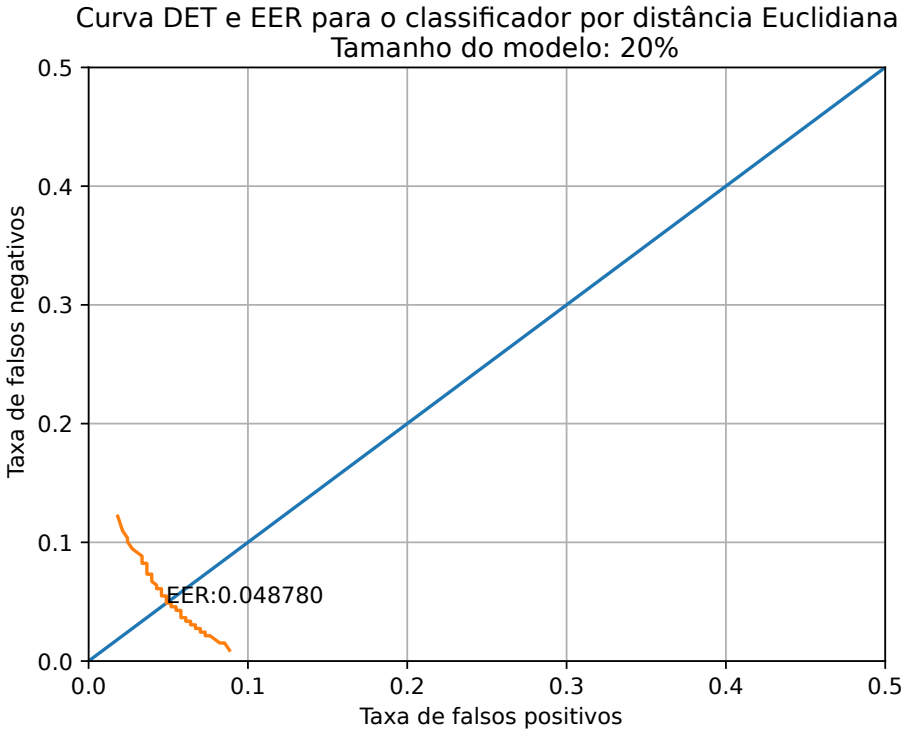
Fonte: Elaborado pelo autor, 2021.

Figura 20 – Acurácia x quantidade de testes - Distância Euclidiana, modelo a 20%



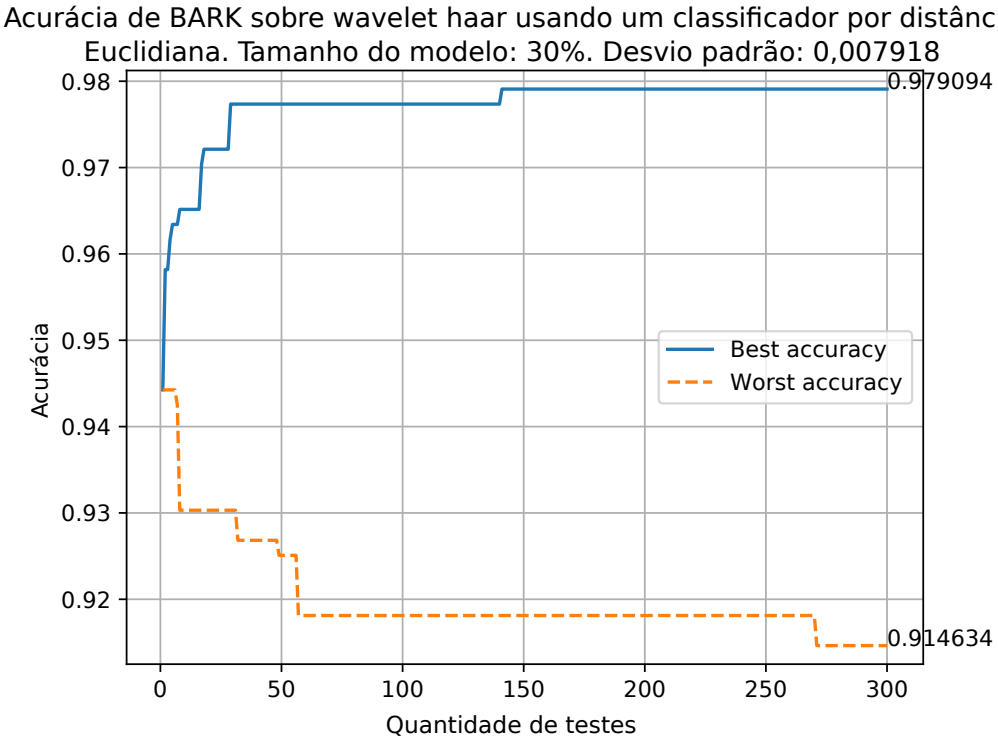
Fonte: Elaborado pelo autor, 2021.

Figura 21 – Curva DET dos resultados de distância Euclidiana, modelo a 20%



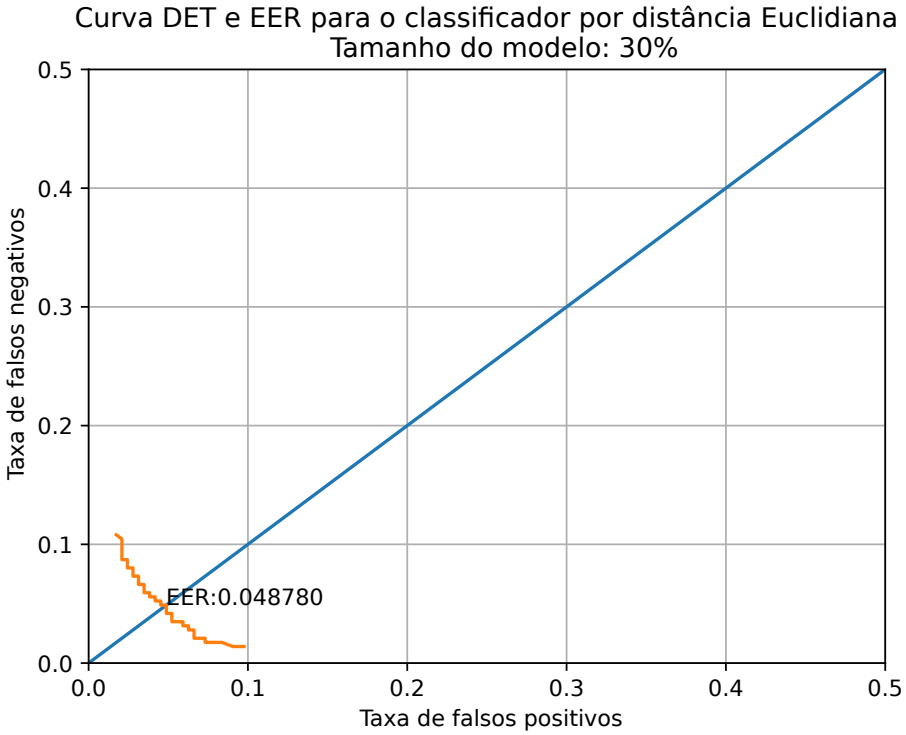
Fonte: Elaborado pelo autor, 2021.

Figura 22 – Acurácia x quantidade de testes - Distância Euclidiana, modelo a 30%



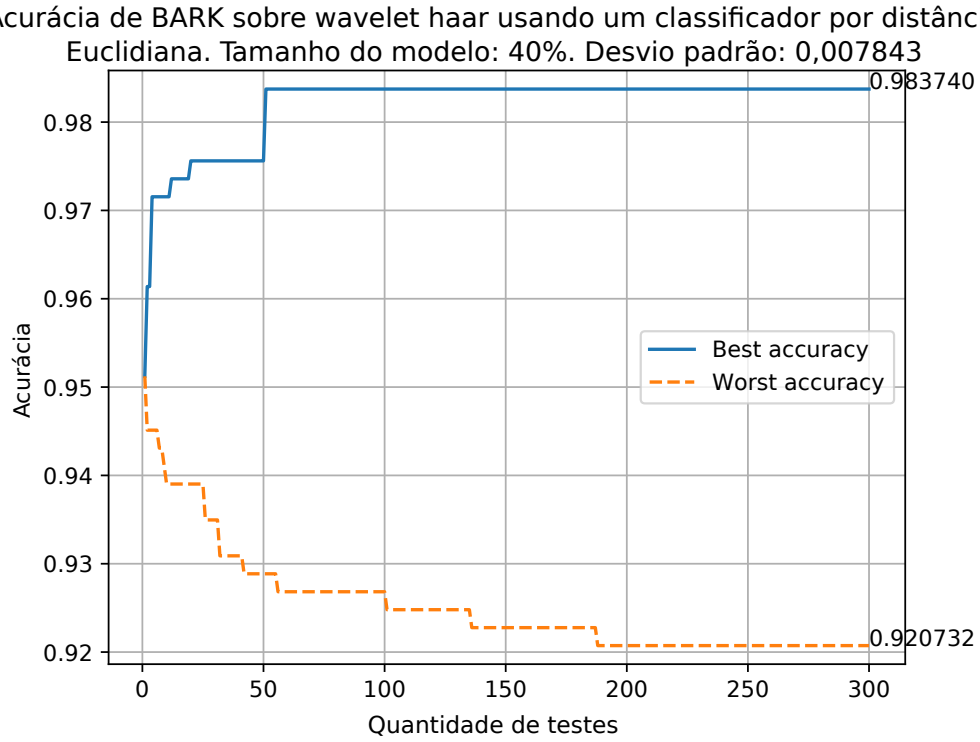
Fonte: Elaborado pelo autor, 2021.

Figura 23 – Curva DET dos resultados de distância Euclidiana, modelo a 30%



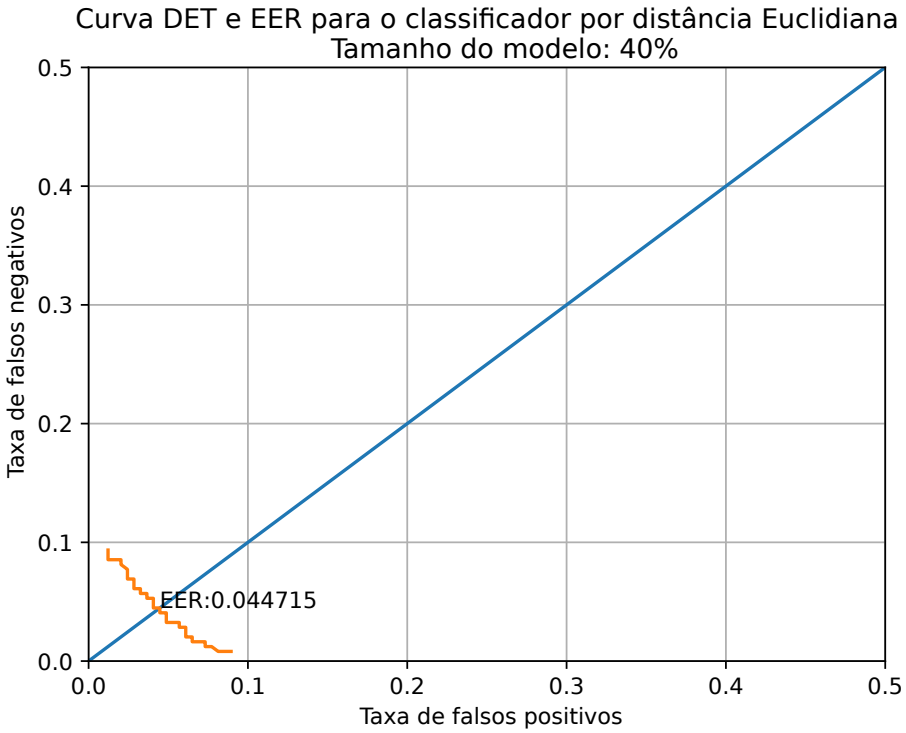
Fonte: Elaborado pelo autor, 2021.

Figura 24 – Acurácia x quantidade de testes - Distância Euclidiana, modelo a 40%



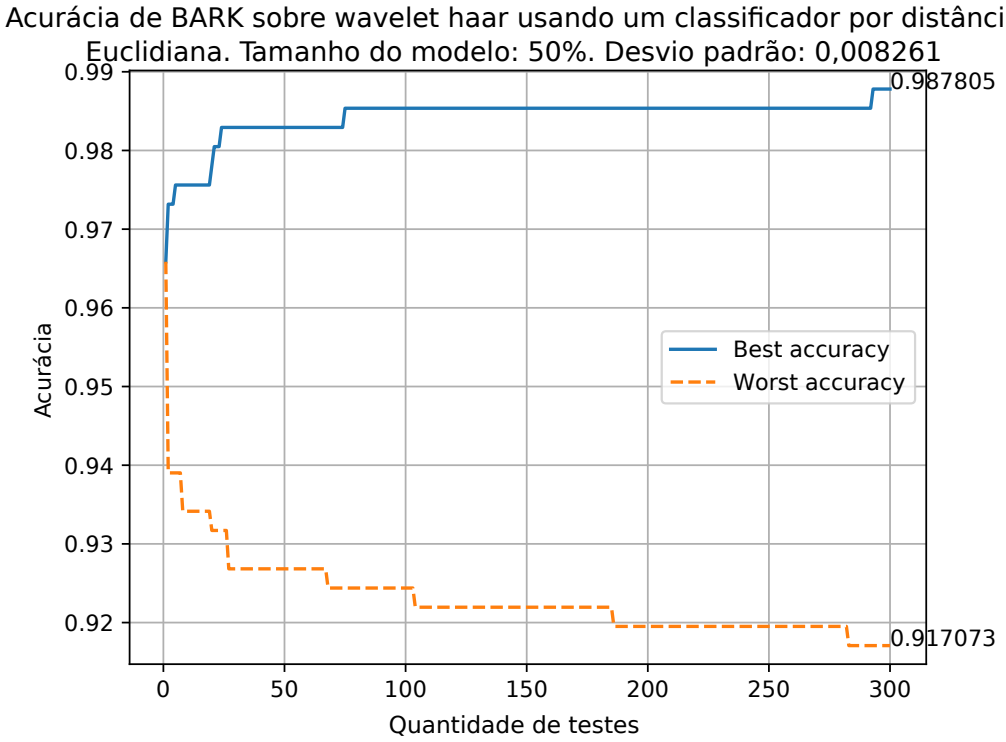
Fonte: Elaborado pelo autor, 2021.

Figura 25 – Curva DET dos resultados de distância Euclidiana, modelo a 40%



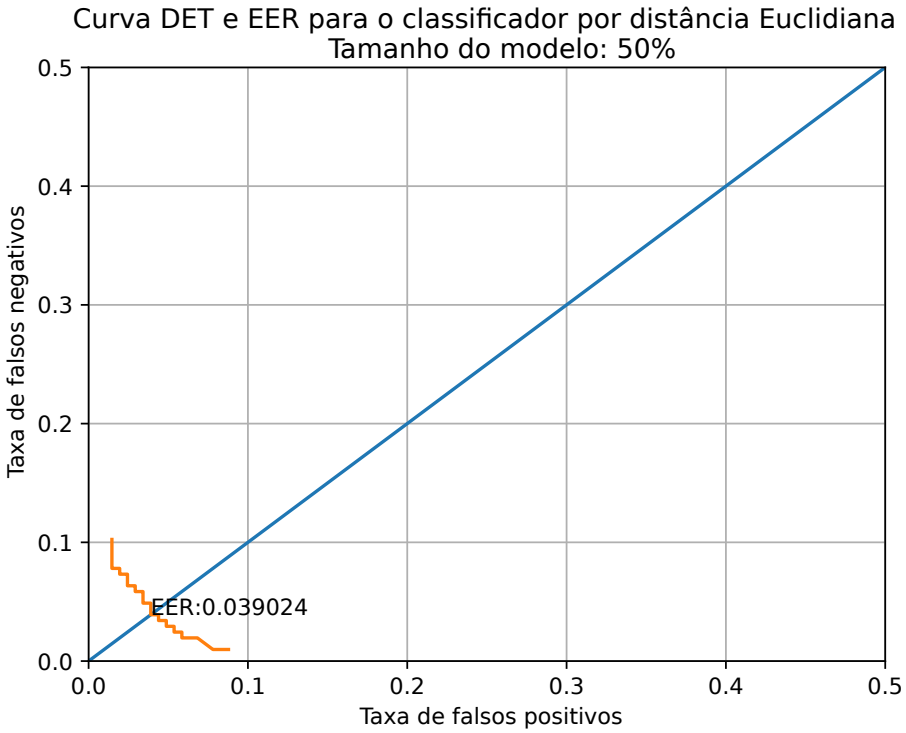
Fonte: Elaborado pelo autor, 2021.

Figura 26 – Acurácia x quantidade de testes - Distância Euclidiana, modelo a 50%



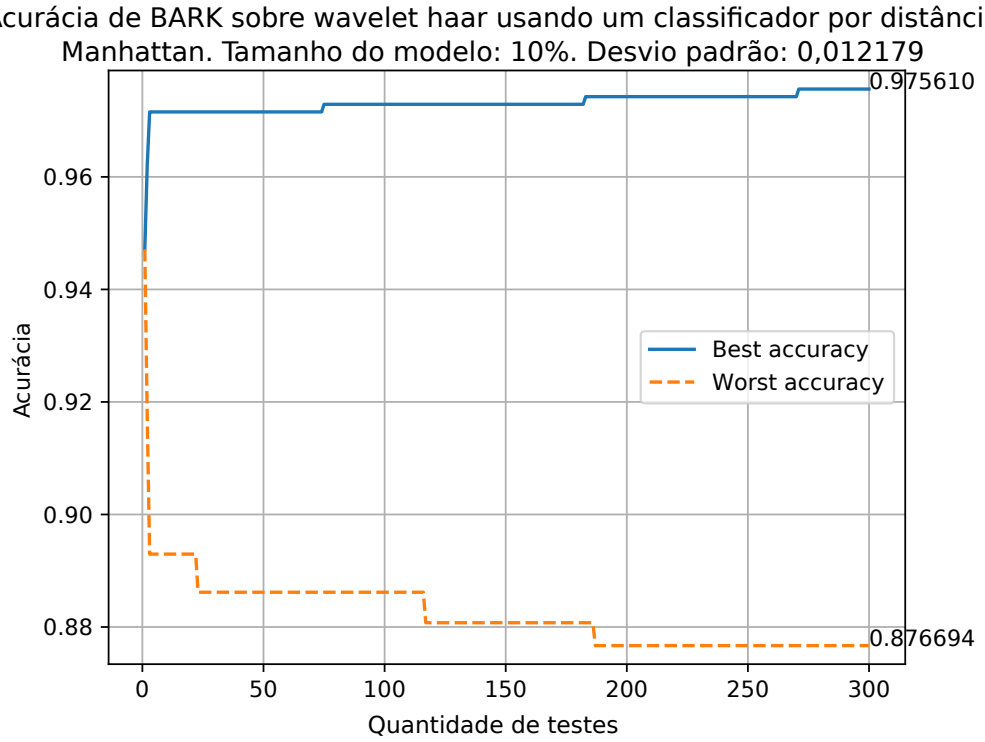
Fonte: Elaborado pelo autor, 2021.

Figura 27 – Curva DET dos resultados de distância Euclidiana, modelo a 50%



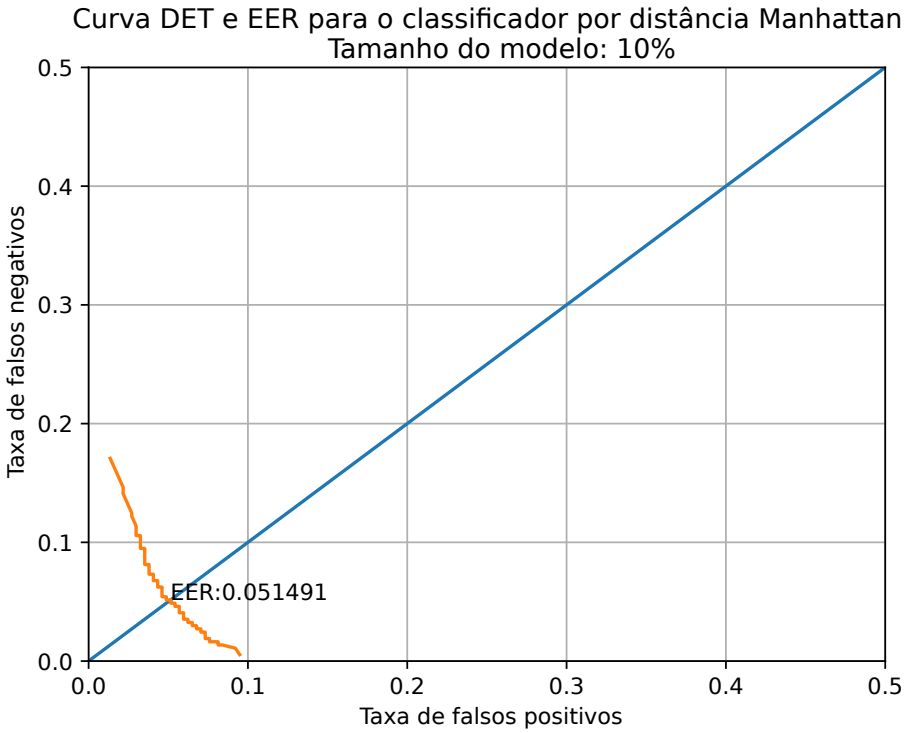
Fonte: Elaborado pelo autor, 2021.

Figura 28 – Acurácia x quantidade de testes - Distância Manhattan, modelo a 10%



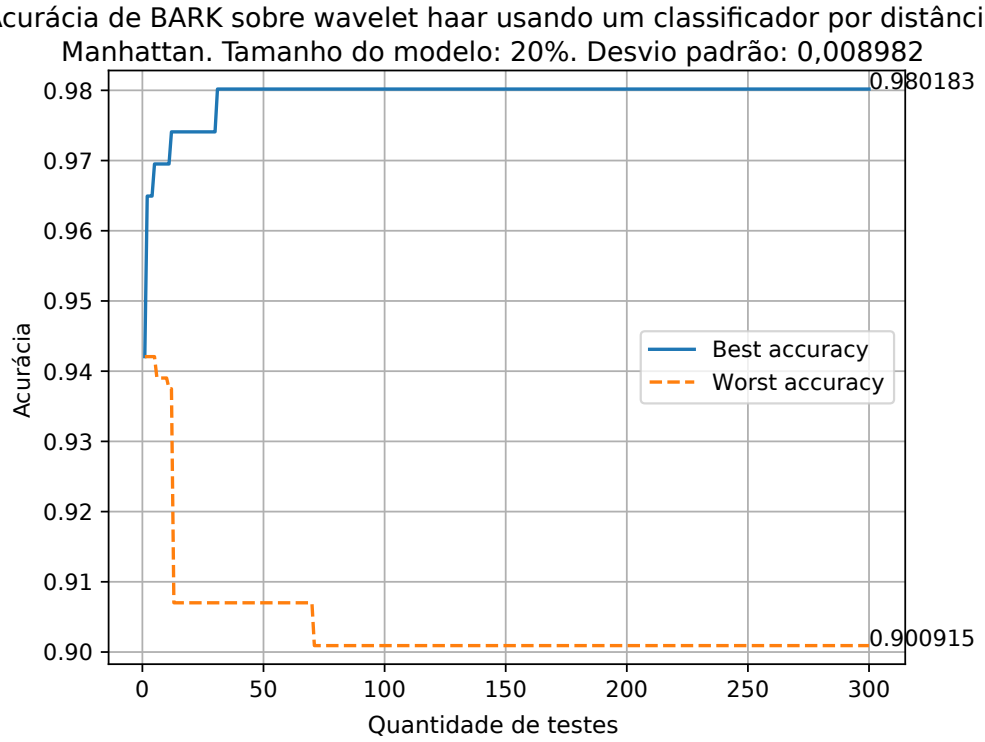
Fonte: Elaborado pelo autor, 2021.

Figura 29 – Curva DET dos resultados de distância Manhattan, modelo a 10%



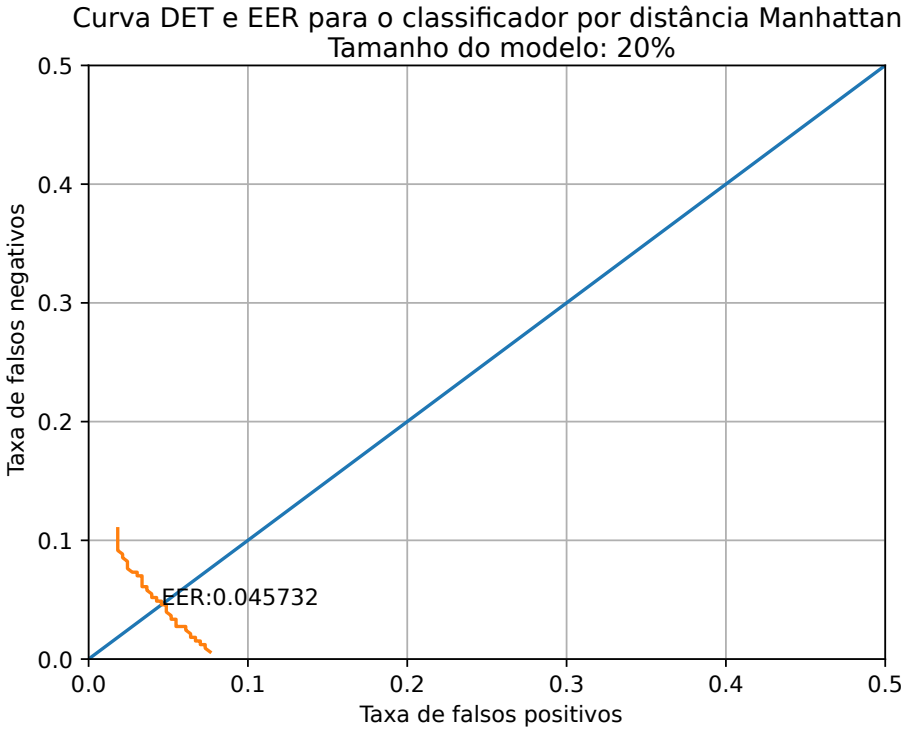
Fonte: Elaborado pelo autor, 2021.

Figura 30 – Acurácia x quantidade de testes - Distância Manhattan, modelo a 20%



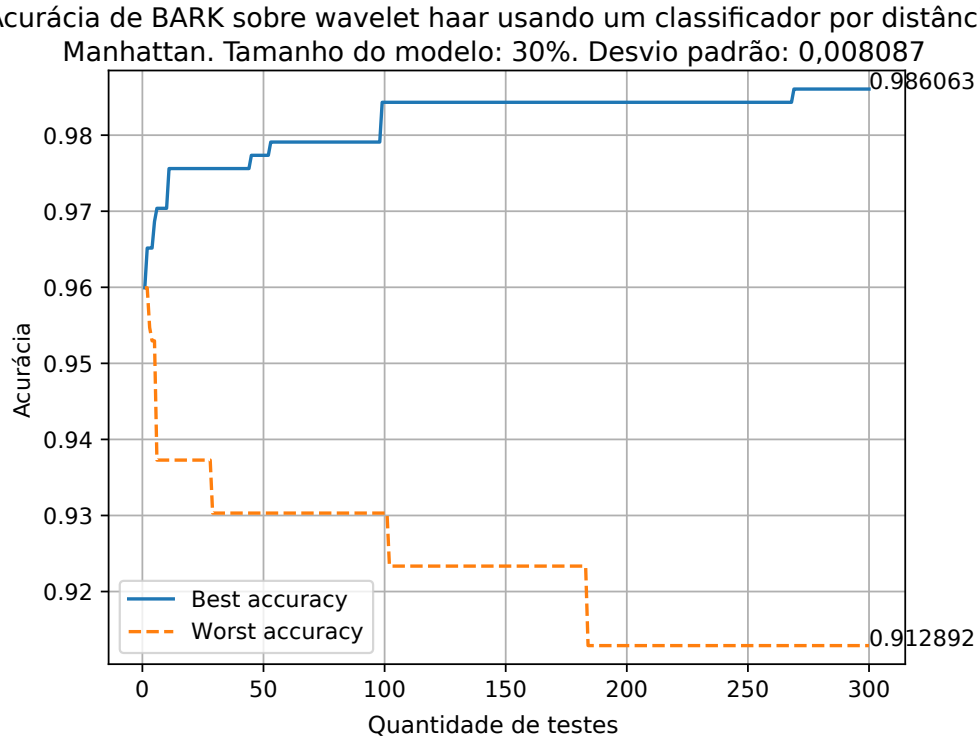
Fonte: Elaborado pelo autor, 2021.

Figura 31 – Curva DET dos resultados de distância Manhattan, modelo a 20%



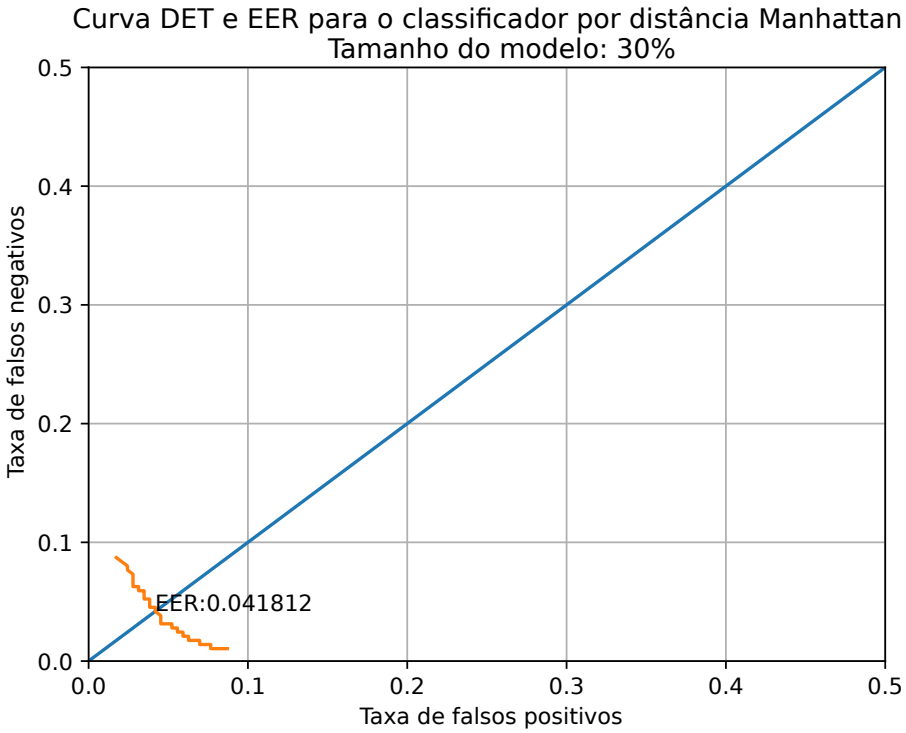
Fonte: Elaborado pelo autor, 2021.

Figura 32 – Acurácia x quantidade de testes - Distância Manhattan, modelo a 30%



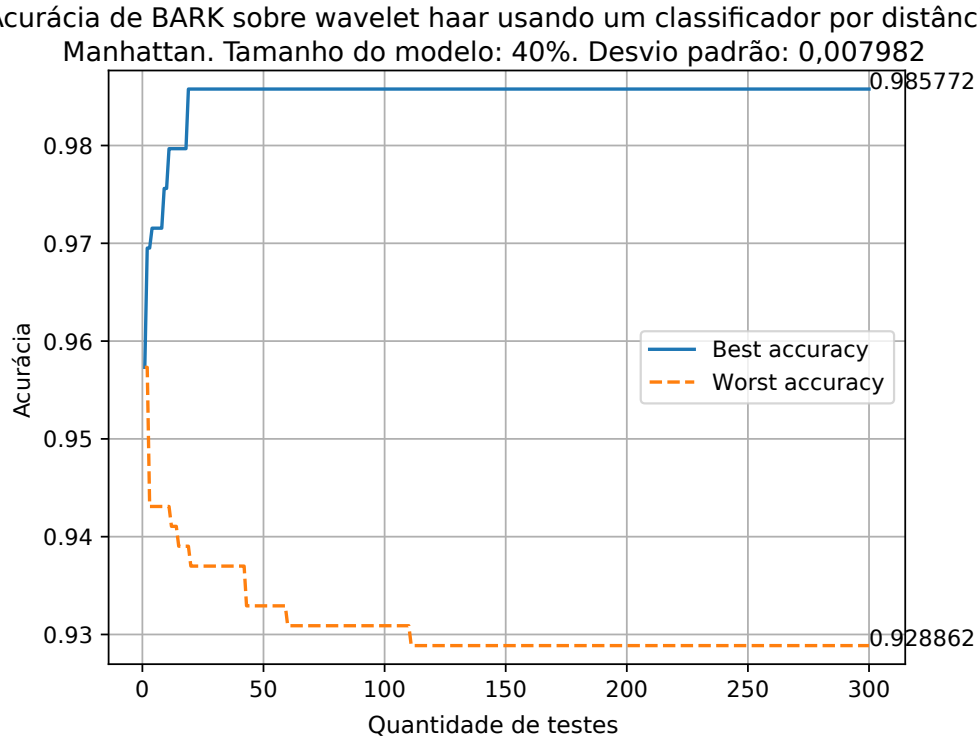
Fonte: Elaborado pelo autor, 2021.

Figura 33 – Curva DET dos resultados de distância Manhattan, modelo a 30%



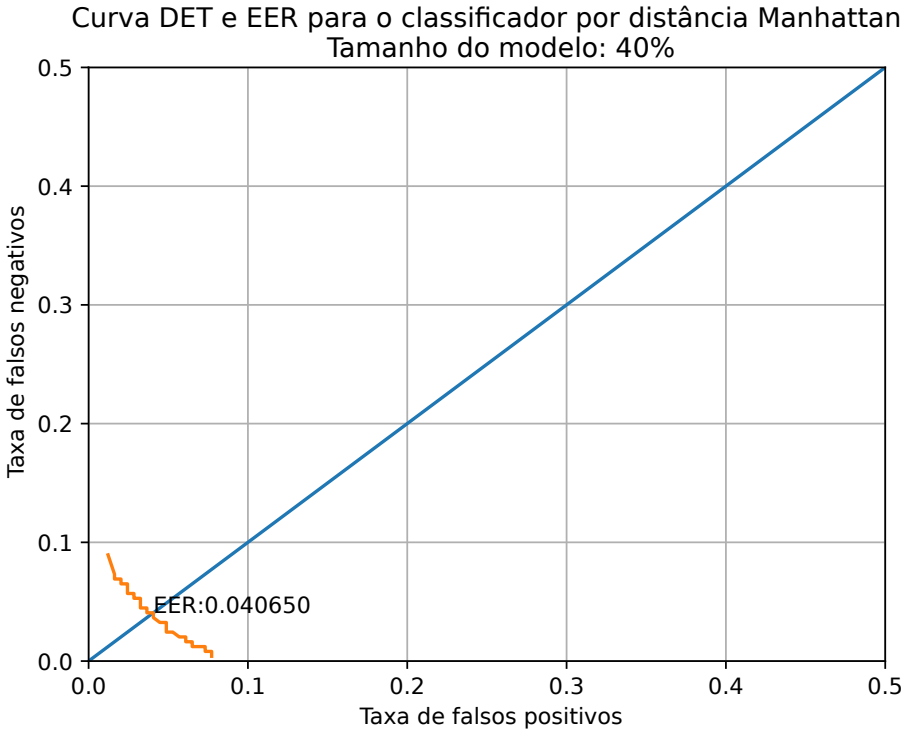
Fonte: Elaborado pelo autor, 2021.

Figura 34 – Acurácia x quantidade de testes - Distância Manhattan, modelo a 40%



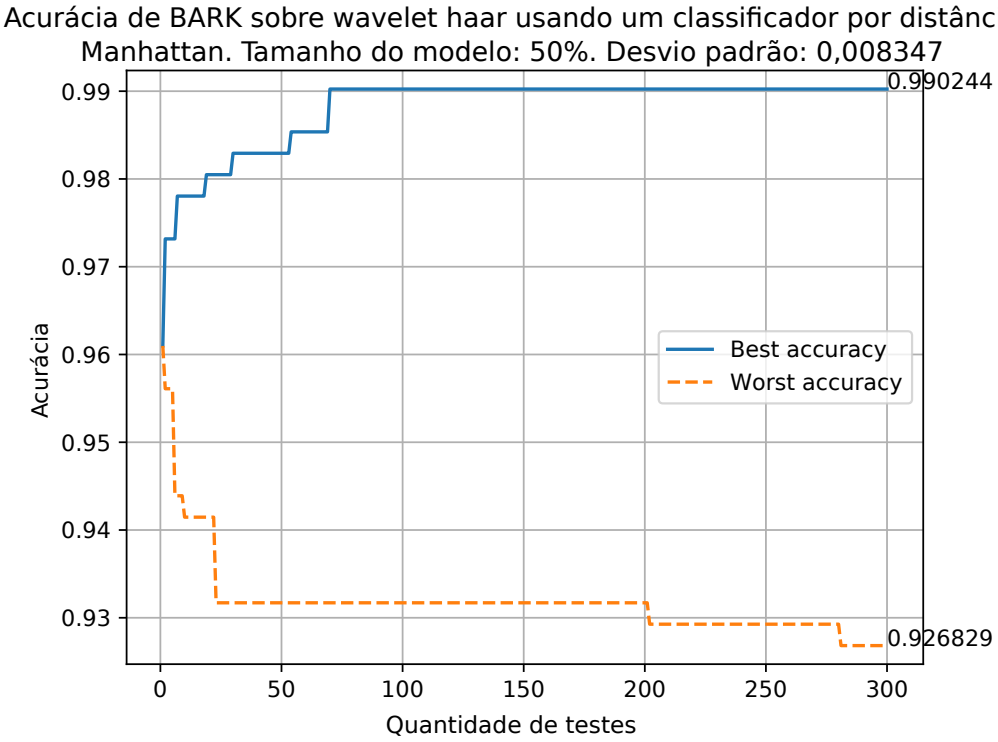
Fonte: Elaborado pelo autor, 2021.

Figura 35 – Curva DET dos resultados de distância Manhattan, modelo a 40%



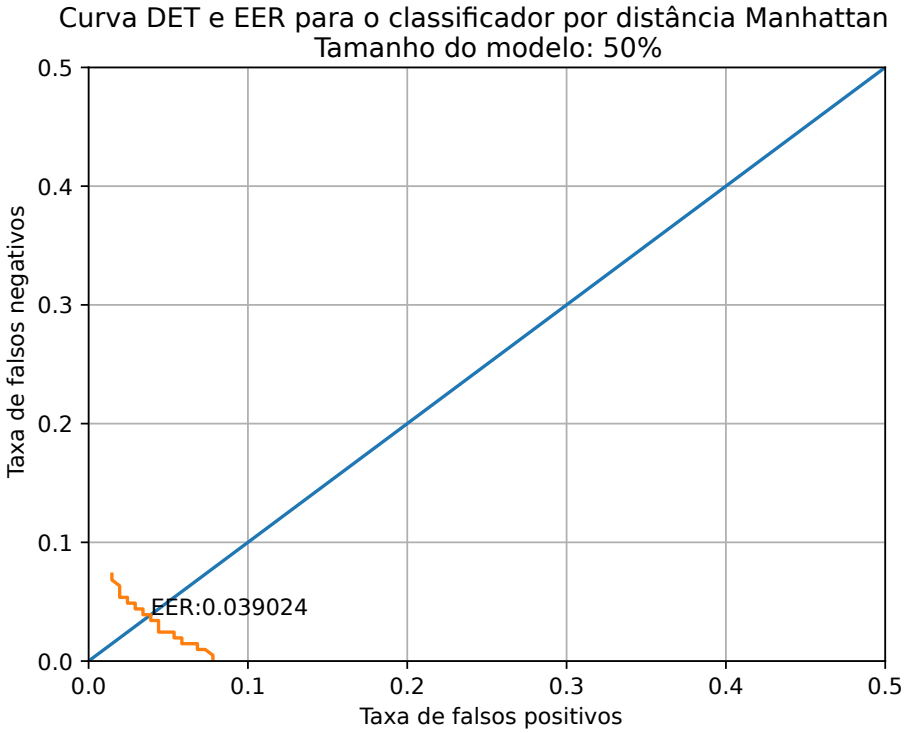
Fonte: Elaborado pelo autor, 2021.

Figura 36 – Acurácia x quantidade de testes - Distância Manhattan, modelo a 50%



Fonte: Elaborado pelo autor, 2021.

Figura 37 – Curva DET dos resultados de distância Manhattan, modelo a 50%



Fonte: Elaborado pelo autor, 2021.

4.2.2 Síntese

Observando as Tabelas, nota-se que as **melhores acurácias/EERs** possuem os valores de **0,987805/0,039024** e **0,990244/0,039024** para as distâncias Euclidiana e Manhattan respectivamente. Desse modo, constata-se uma pequena diferença prática no critério de métrica *pattern-matching* utilizado e, além disso, um nível de acurácia relativamente alto considerando-se a simplicidade do algoritmo classificador.

4.3 Procedimento 03

Novamente, considerando que o procedimento 01 apresentou, como melhor resultado, a combinação **Haar + Bark**, o objetivo deste procedimento é constatar a **máxima acurácia e mínimo EER** que se consegue atingir com uma SVM. Os resultados, obedecendo os mesmo moldes do procedimento anterior, constam na Tabela 19.

Mais níveis de detalhes podem ser consultados nas Tabelas 20a, 20b, 21a, 21b, 22a, 22b, 23a, 23b, 24a e 24b, além dos seus respectivos gráficos nas Figuras 38, 40, 42, 44 e 46.

Tabela 19 – Resultados da abordagem com SVM

M	**Acurácia mínima**	**Acurácia máxima**	**Média das acurácias**	**Desvio padrão da acurácia**	**EER**
10%	0,872629	0,982385	0,927507	0,008591	0,04336
20%	0,916159	0,98628	0,9512195	0,0071	0,036585
30%	0,93554	0,993031	0,9642855	0,006916	0,031359
40%	0,936992	0,993902	0,965447	0,007044	0,028455
50%	0,936585	0,997561	0,967073	0,00739	0,02439

Fonte: Elaborado pelo autor, 2021.

As tabelas de confusão com os respectivos gráficos foram agrupados em uma mesma página para melhorar a leitura e o entendimento.

4.3.1 Resultados para BARK com *wavelet* Haar usando um classificador SVM

Tabela 20 – Matrizes de confusão para SVM com modelo a 10%

(a) Melhor matriz de confusão

	genuíno	falseado
genuíno	366	10
falseado	3	359

(b) Pior matriz de confusão

	genuíno	falseado
genuíno	284	9
falseado	85	360

Fonte: Elaborado pelo autor, 2021.

Tabela 21 – Matrizes de confusão para SVM com modelo a 20%

(a) Melhor matriz de confusão

	genuíno	falseado
genuíno	324	5
falseado	4	323

(b) Pior matriz de confusão

	genuíno	falseado
genuíno	289	16
falseado	39	312

Fonte: Elaborado pelo autor, 2021.

Tabela 22 – Matrizes de confusão para SVM com modelo a 30%

(a) Melhor matriz de confusão

	genuíno	falseado
genuíno	287	4
falseado	0	283

(b) Pior matriz de confusão

	genuíno	falseado
genuíno	265	15
falseado	22	272

Fonte: Elaborado pelo autor, 2021.

Tabela 23 – Matrizes de confusão para SVM com modelo a 40%

(a) Melhor matriz de confusão

	genuíno	falseado
genuíno	243	0
falseado	3	246

(b) Pior matriz de confusão

	genuíno	falseado
genuíno	226	11
falseado	20	235

Fonte: Elaborado pelo autor, 2021.

Tabela 24 – Matrizes de confusão para SVM com modelo a 50%

(a) Melhor matriz de confusão

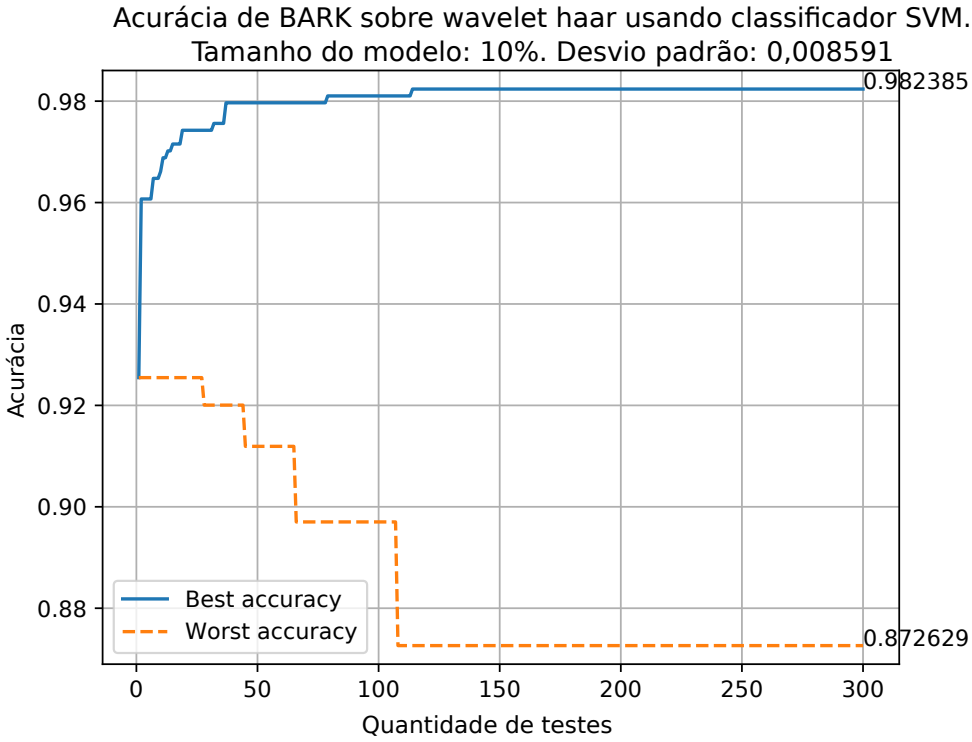
	genuíno	falseado
genuíno	205	1
falseado	0	204

(b) Pior matriz de confusão

	genuíno	falseado
genuíno	196	17
falseado	9	188

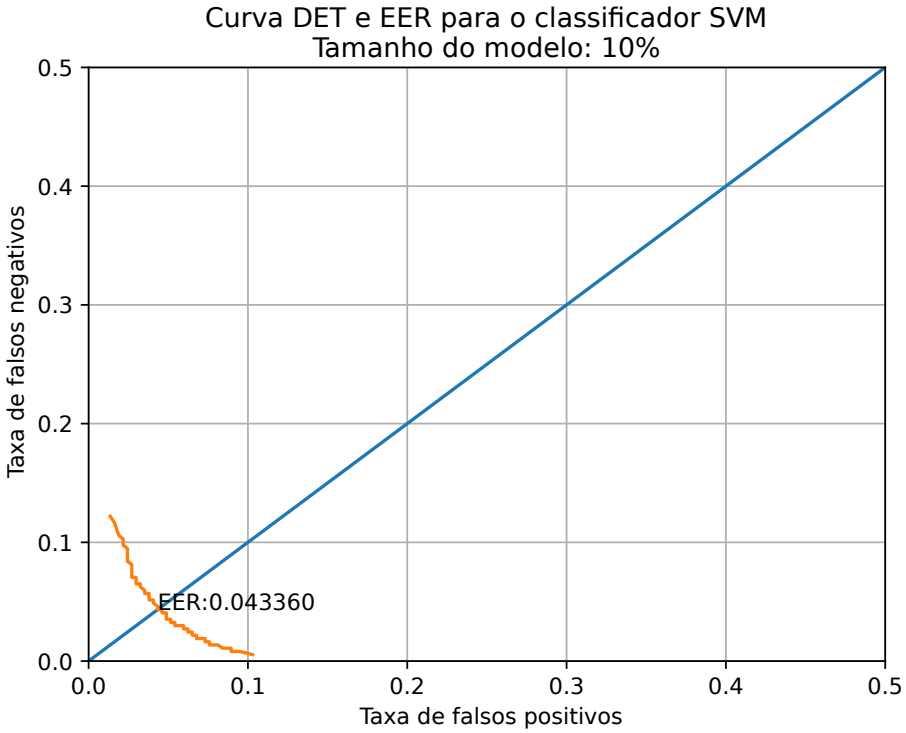
Fonte: Elaborado pelo autor, 2021.

Figura 38 – Acurácia x quantidade de testes - SVM, modelo a 10%



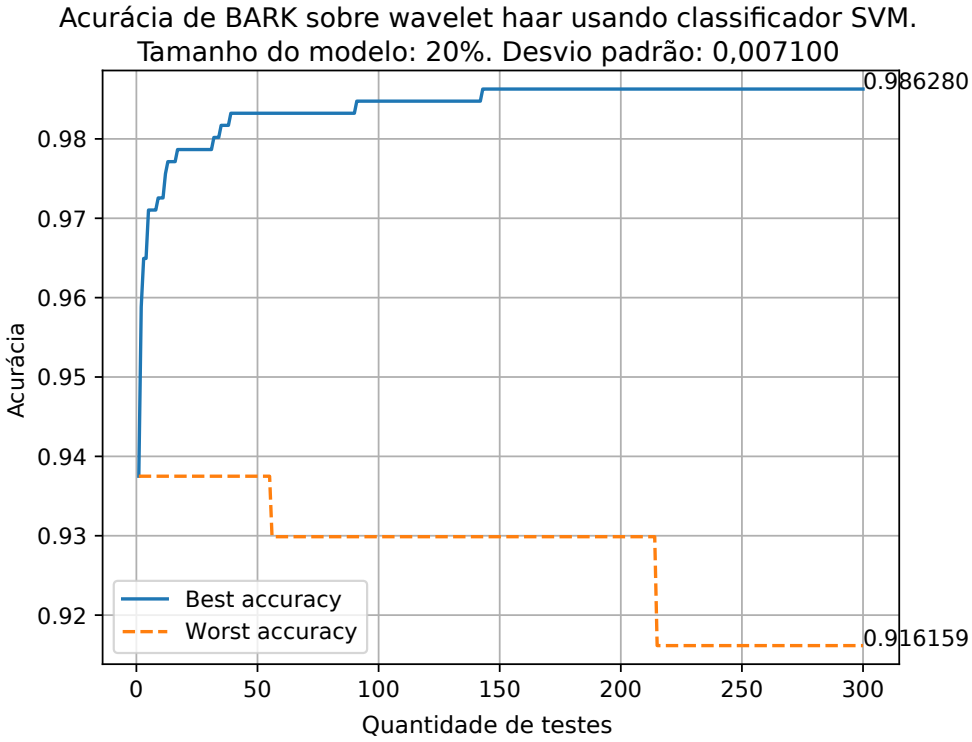
Fonte: Elaborado pelo autor, 2021.

Figura 39 – Curva DET dos resultados da SVM, modelo a 10%



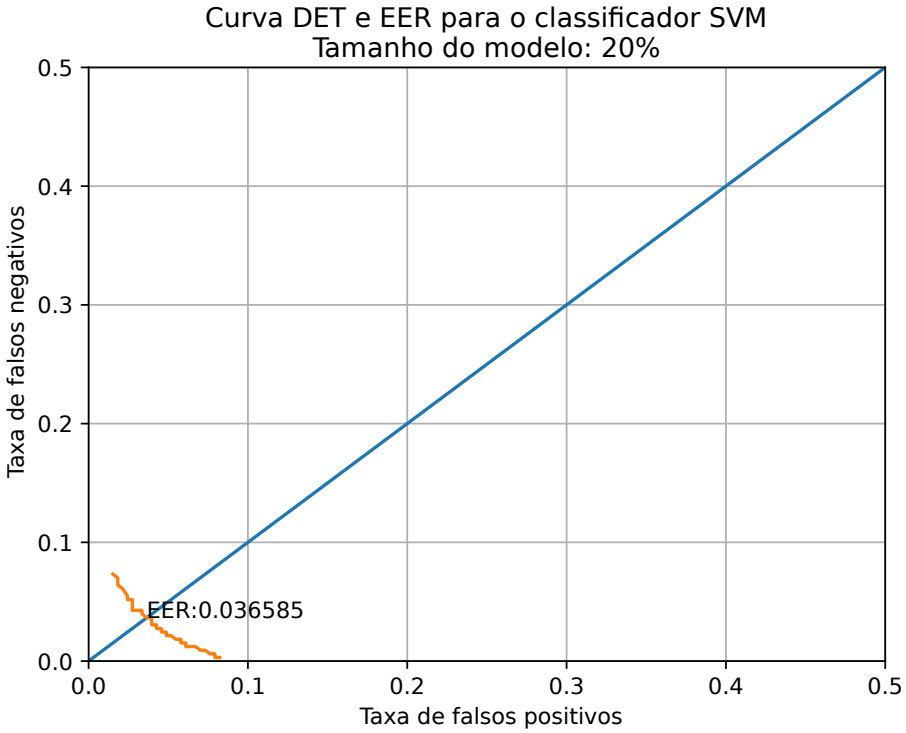
Fonte: Elaborado pelo autor, 2021.

Figura 40 – Acurácia x quantidade de testes - SVM, modelo a 20%



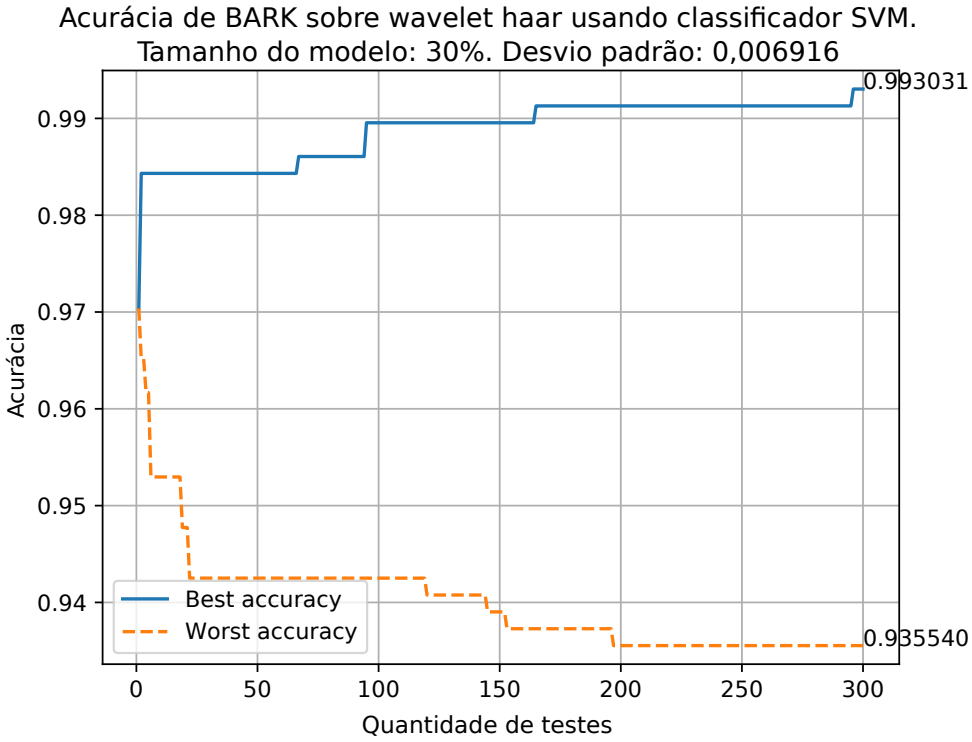
Fonte: Elaborado pelo autor, 2021.

Figura 41 – Curva DET dos resultados da SVM, modelo a 20%



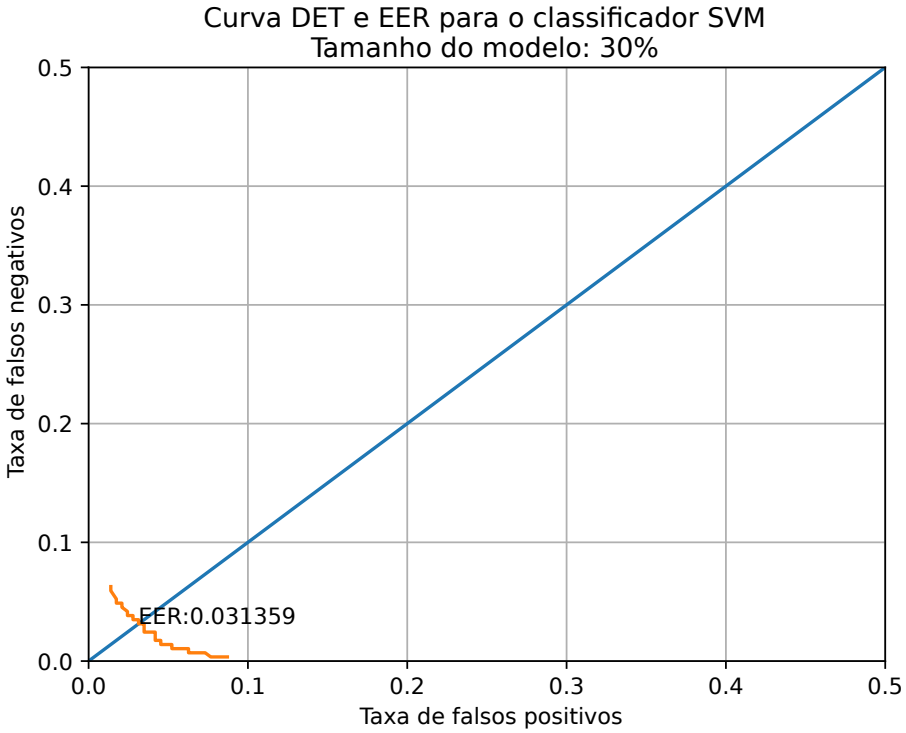
Fonte: Elaborado pelo autor, 2021.

Figura 42 – Acurácia x quantidade de testes - SVM, modelo a 30%



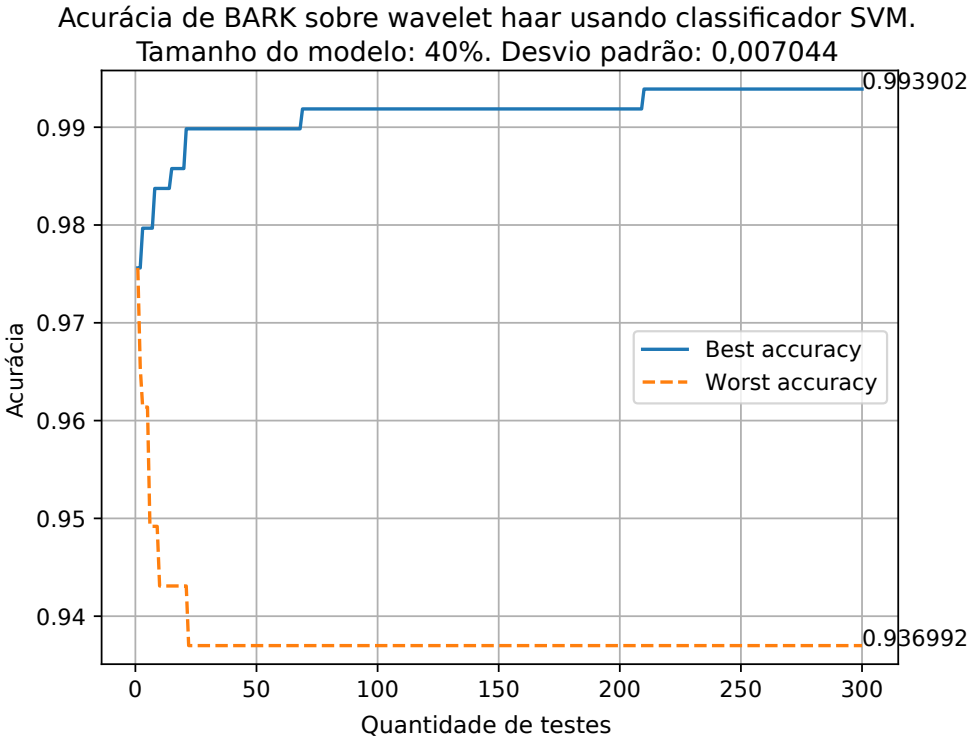
Fonte: Elaborado pelo autor, 2021.

Figura 43 – Curva DET dos resultados da SVM, modelo a 30%



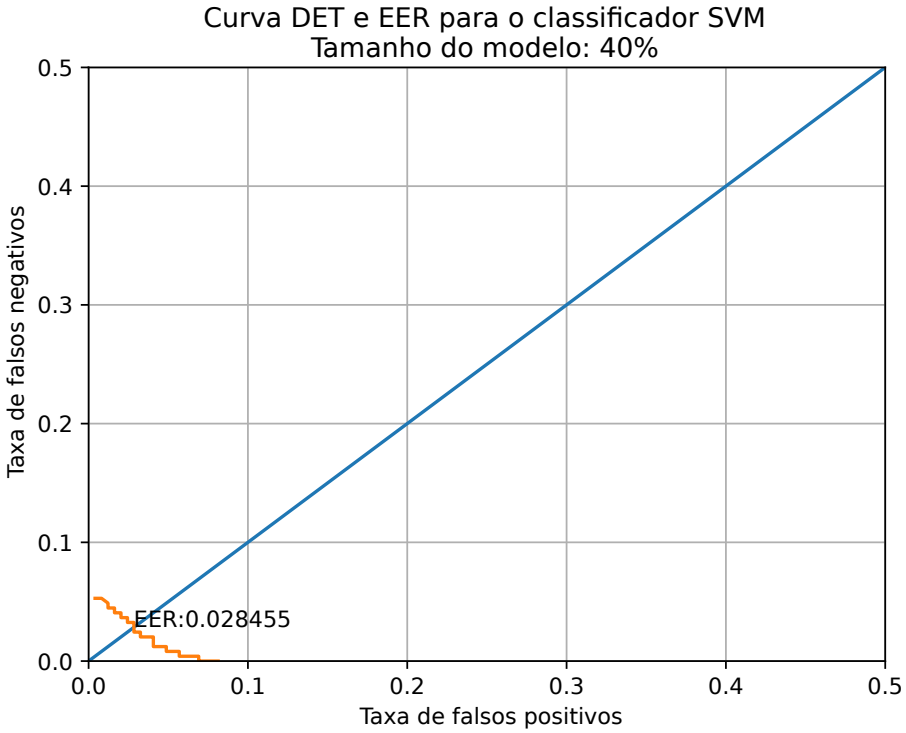
Fonte: Elaborado pelo autor, 2021.

Figura 44 – Acurácia x quantidade de testes - SVM, modelo a 40%



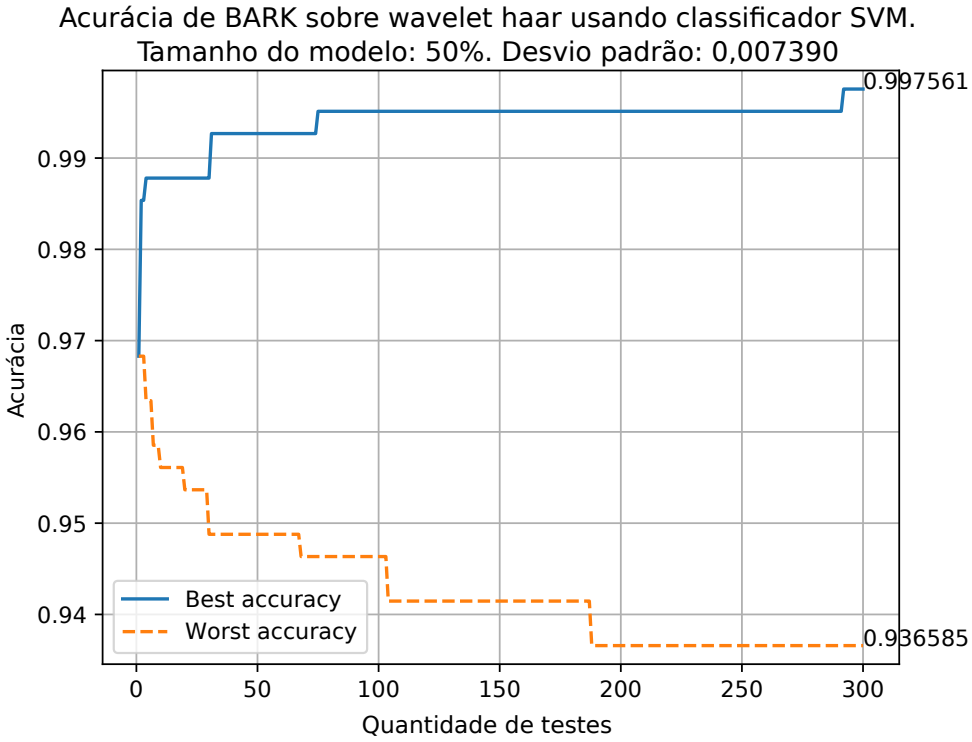
Fonte: Elaborado pelo autor, 2021.

Figura 45 – Curva DET dos resultados da SVM, modelo a 40%



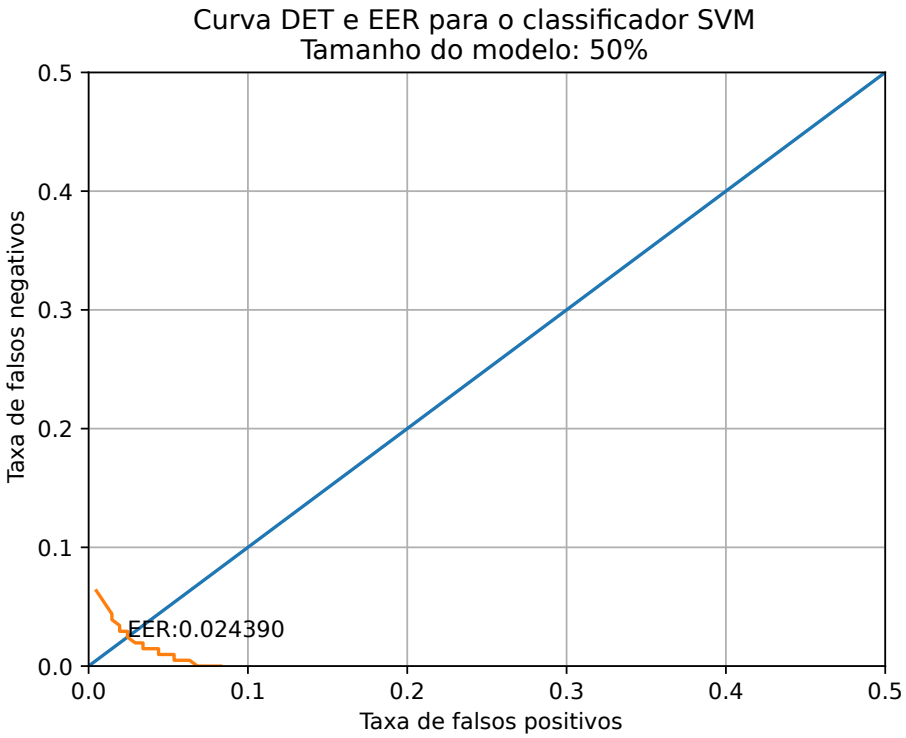
Fonte: Elaborado pelo autor, 2021.

Figura 46 – Acurácia X quantidade de testes - SVM, modelo a 50%



Fonte: Elaborado pelo autor, 2021.

Figura 47 – Curva DET dos resultados da SVM, modelo a 50%



Fonte: Elaborado pelo autor, 2021.

4.3.2 Síntese

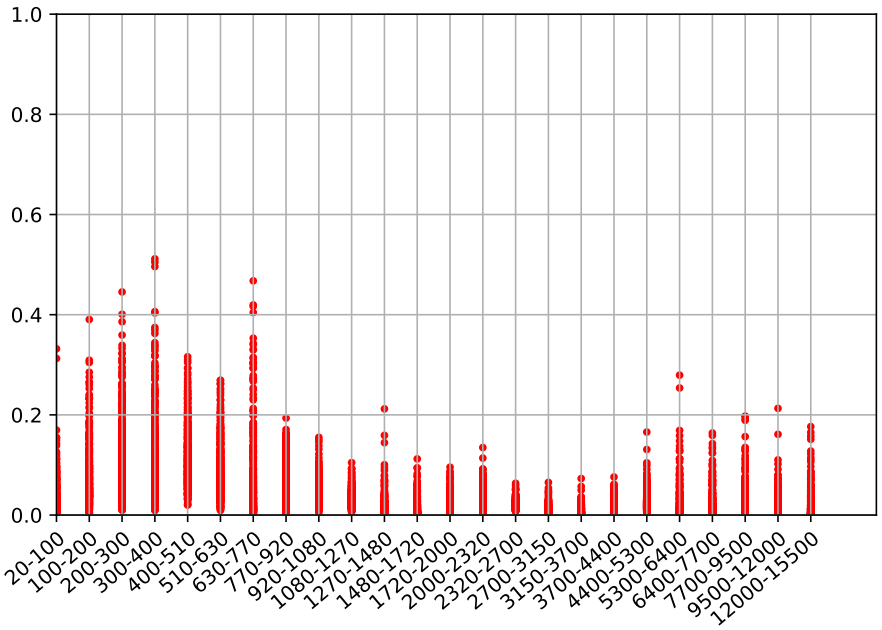
Em suma, percebe-se que a **melhor acurácia/EER** obtida foi de **0,997561/0,02439** com o uso da SVM e 50% da base de sinais para treiná-la, condizente com as expectativas. Em tal caso, a respectiva matriz de confusão revela uma quantidade mínima de falsos-genuínos e falsos-falseados, condizentes com os trabalhos correlatos revisados anteriormente.

4.4 Testes Complementares

As figuras 48 e 49 contém, como complemento, o espalhamento de valores dos vetores de características na *wavelet-packet* Haar + escala Bark. Notavelmente, comparando-se com as Figuras 50 e 51, nota-se que **na escala *Mel* os vetores de características para os sinais falseados e genuínos são mais similares entre si do que na escala *Bark***. Essa diferença na distribuição dos dados também ocorre nas combinações *wavelet-packet* daubechies 76 + Bark x daubechies 76 + Mel conforme as Figuras 52, 53 e 54, 55.

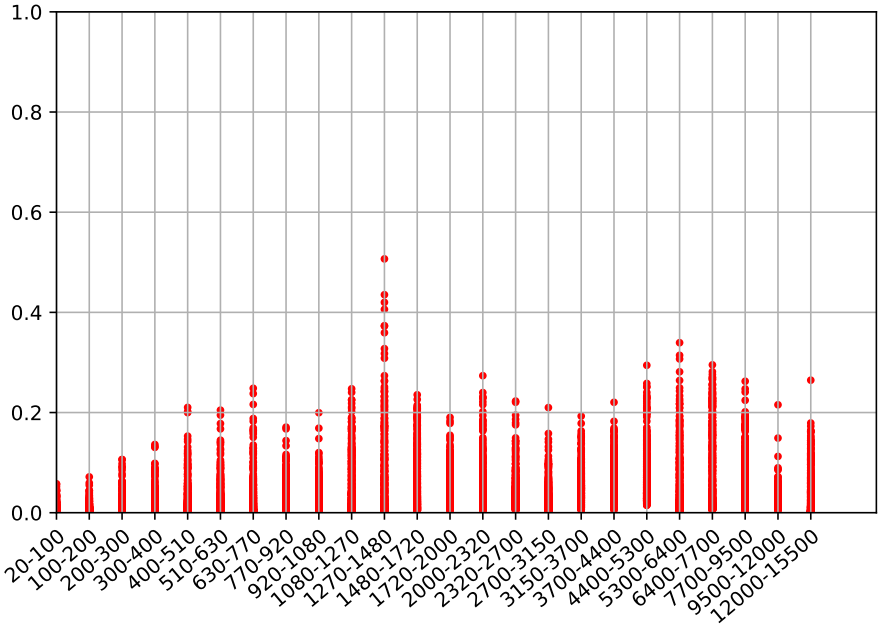
Para fins de comparação, os gráficos foram colocados em uma mesma escala no eixo das amplitudes (vertical).

Figura 48 – Gráfico de dispersão para os vetores de características *genuínos* obtidos com *Haar + BARK*. Eixos horizontais e verticais: Bandas de frequência em Hertz(Hz) e amplitudes, respectivamente.



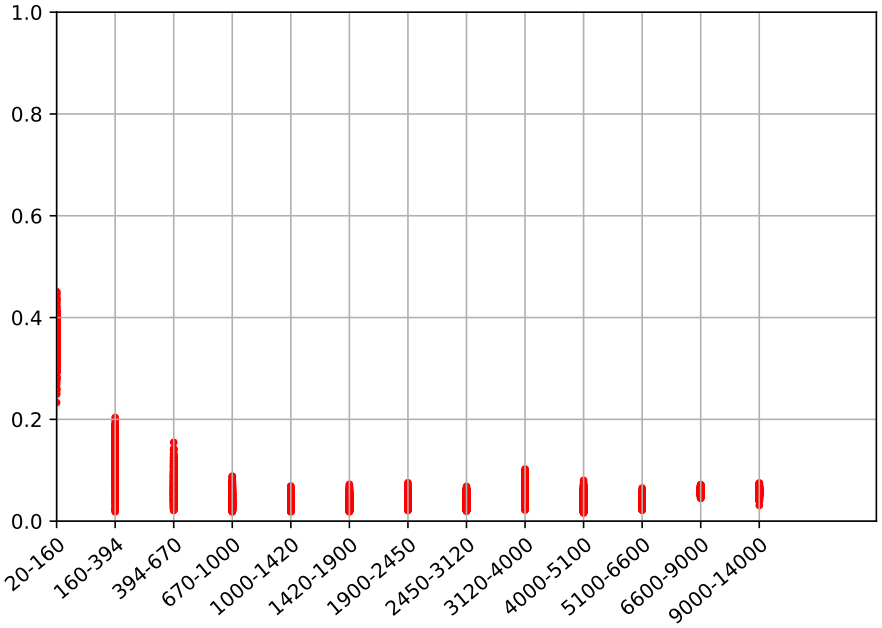
Fonte: Elaborado pelo autor, 2021.

Figura 49 – Gráfico de dispersão para os vetores de características *falseados* obtidos com *Haar + BARK*. Eixos horizontais e verticais: Banda de frequência em Hertz(Hz) e amplitudes, respectivamente.



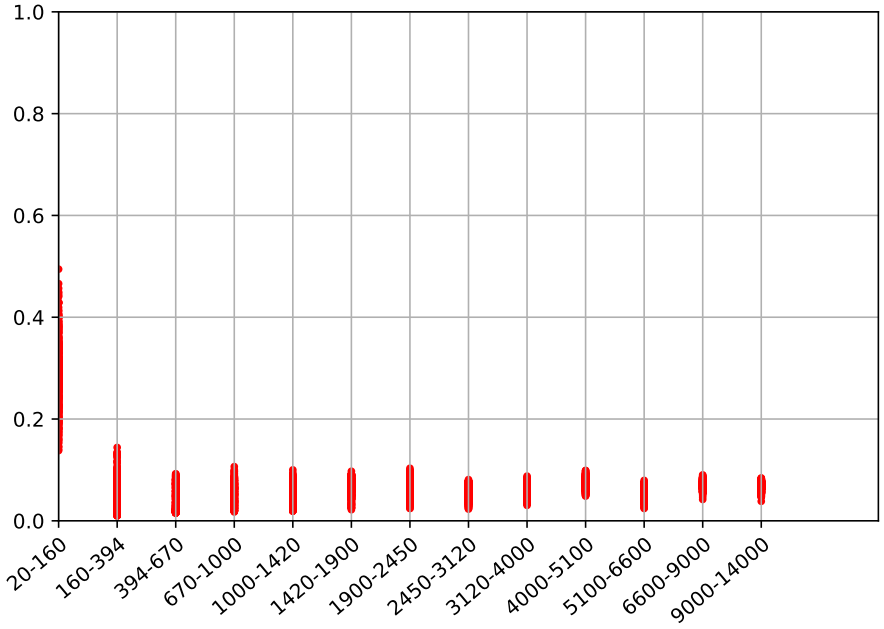
Fonte: Elaborado pelo autor, 2021.

Figura 50 – Gráfico de dispersão para vetores de características *genuínos* obtidos com *Haar + MEL*. Eixos horizontais e verticais: Bandas de frequência em Hertz(Hz) e amplitudes, respectivamente.



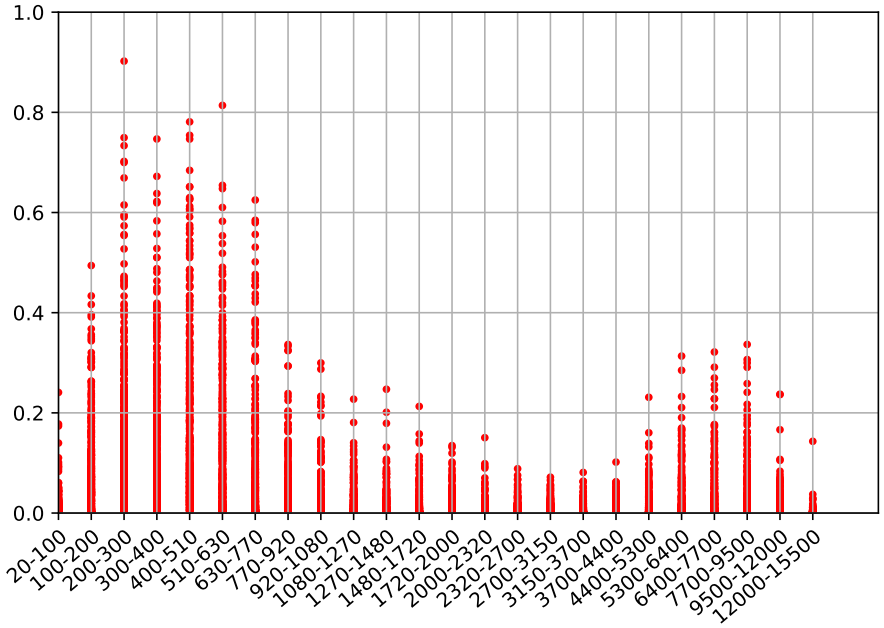
Fonte: Elaborado pelo autor, 2021.

Figura 51 – Gráfico de dispersão para vetores de características *falseados* obtidos com *Haar + MEL*. Eixos horizontais e verticais: Bandas de frequência em Hertz(Hz) e amplitudes, respectivamente.



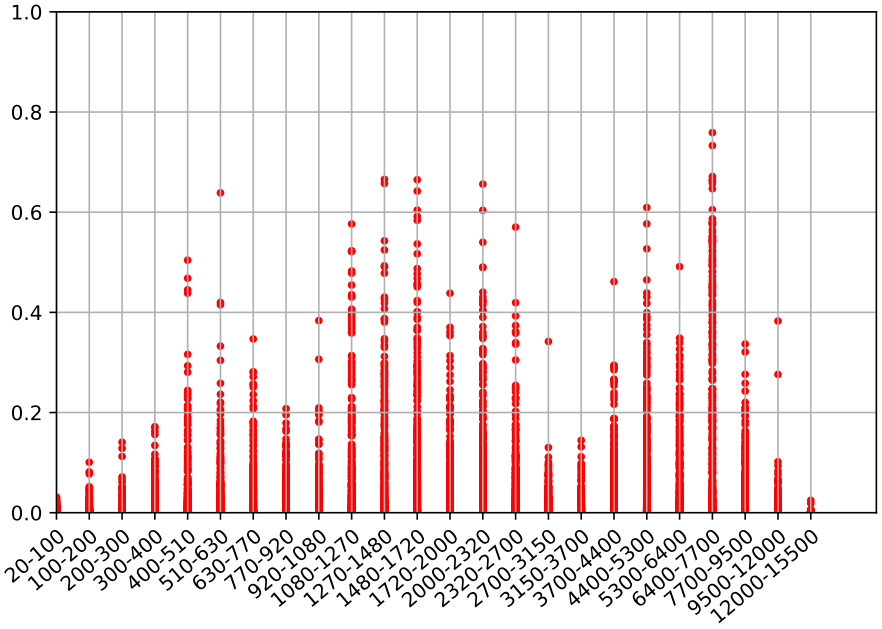
Fonte: Elaborado pelo autor, 2021.

Figura 52 – Gráfico de dispersão para vetores de características *genuínos* obtidos com *daubechies 76 + BARK*. Eixos horizontais e verticais: Bandas de frequência em Hertz(Hz) e amplitudes, respectivamente.



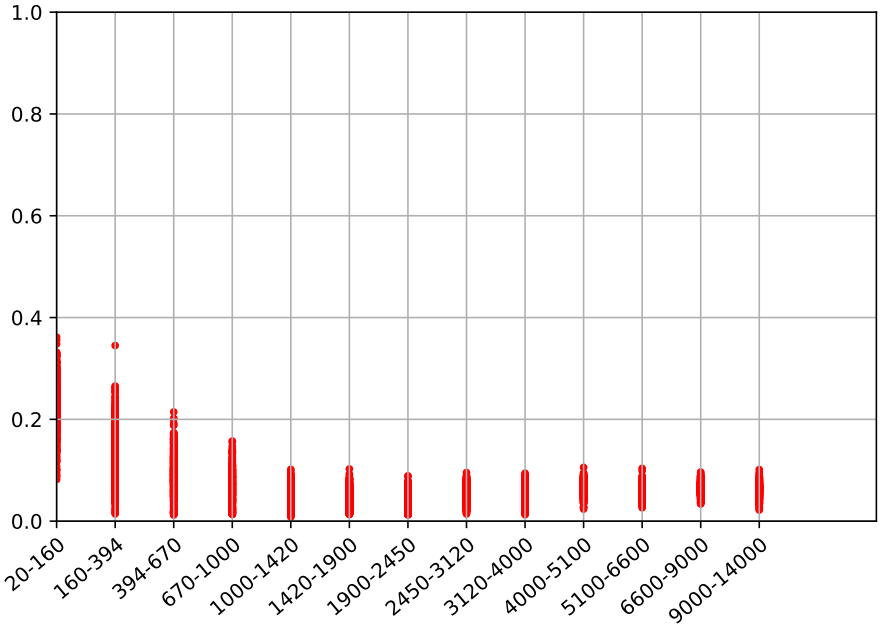
Fonte: Elaborado pelo autor, 2021.

Figura 53 – Gráfico de dispersão para vetores de características *falseados* obtidos com *daubechies 76 + BARK*. Eixos horizontais e verticais: Bandas de frequência em Hertz(Hz) e amplitudes, respectivamente.



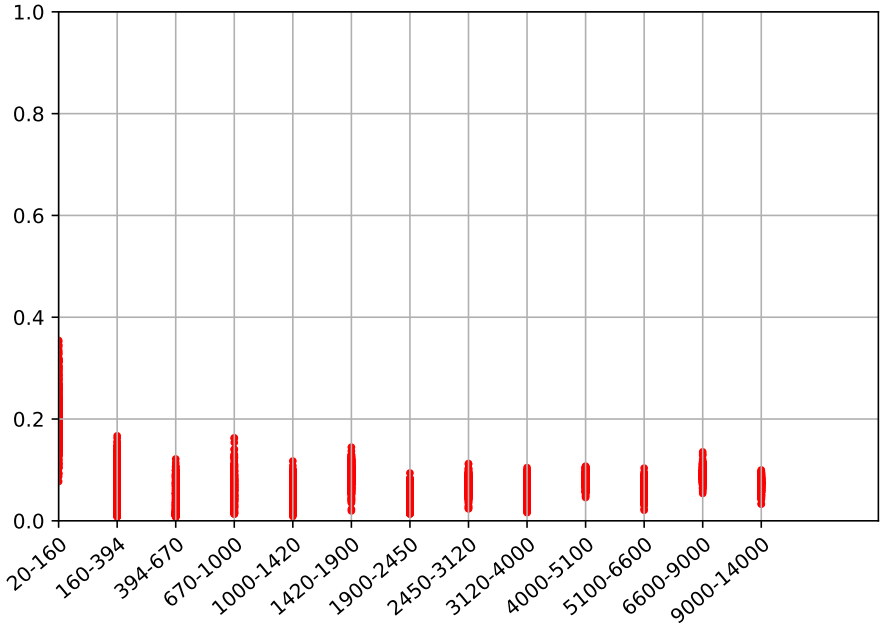
Fonte: Elaborado pelo autor, 2021.

Figura 54 – Gráfico de dispersão para vetores de características *genuínos* obtidos com *daubechies 76 + MEL*. Eixos horizontais e verticais: Bandas de frequência em Hertz(Hz) e amplitudes, respectivamente.



Fonte: Elaborado pelo autor, 2021.

Figura 55 – Gráfico de dispersão para vetores de características *falseados* obtidos com *daubechies 76 + MEL*. Eixos horizontais e verticais: Bandas de frequência em Hertz(Hz) e amplitudes, respectivamente.



Fonte: Elaborado pelo autor, 2021.

5 Conclusões e Trabalhos Futuros

As hipóteses iniciais deste trabalho consideravam que, a decomposição dos sinais de voz no máximo nível possível, usando *wavelets* de qualquer natureza, não alteraria o resultado quando do cálculo de energia do sinal segundo as escalas *BARK* ou *MEL* já que, no final, os intervalos das frequências seriam, de qualquer forma, tratados segundo as regras de cada escala. Após tal fato mostrar-se falso, como constatado nos resultados, que demonstraram ser a combinação ***wavelet-packet Haar + escala Bark*** a melhor posicionada no plano paraconsistente e *wavelet-packet daubechies 8 + escala Mel* a pior, uma outra hipótese se formou.

A segunda hipótese inicial era que as combinações *wavelet + Mel* seriam superiores a *wavelet + Bark*, pois a primeira é otimizada segundo a percepção humana das frequências sonoras. Novamente, esse pensamento se provou incorreto e inconsistente. Analisando os resultados, é possível notar que, em geral, a escala ***Bark* é superior a *MEL* por promover uma diferenciação maior entre os vetores de características dos sinais genuínos e falseados**, tornando as diferenças das energias calculadas entre as classes, genuína ou falseada, maiores, possibilitando que os vetores de características interclasses sejam mais facilmente diferenciáveis.

De modo geral, os experimentos mostraram que, **com ajuda da engenharia paraconsistente de características é possível utilizar classificadores relativamente modestos já que essa metodologia fornece os vetores de características que representam mais propriamente as classes**.

Em se tratando de uma continuação deste trabalho, a intenção de se explorar os resultados das transformadas *wavelets* até um certo nível somente, em oposição a transformações sucessivas como as que foram utilizadas neste trabalho, ao menos hipoteticamente, parece promissora. Aplicar *wavelets* de forma que fique clara a separação dos componentes de alta e baixa frequência do sinal aparenta ser uma boa base para verificação de *voice spoofing*, pois, quando os gráficos representativos para cada nível de transformação são observados, ficam visualmente nítidas as diferenças entre o sinal genuíno e o falseado. No tocante a utilização de escalas e considerando que o objetivo da proposta seja apenas verificar a falsificação ou não de um sinal vocal, novamente hipoteticamente, talvez fosse mais interessante focar a análise em bandas de frequência superiores dado que, durante os estudos dos sinais falseados, os mesmos apareciam "contaminados" com componentes de alta frequência.

Referências

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.

ADIGA, H. Writing endian-independent code in c. *Retrieved April*, v. 24, 2007.

ALLURI, K. R.; VUPPALA, A. K. Replay spoofing countermeasures using high spectro-temporal resolution features. *International Journal of Speech Technology*, Springer, v. 22, n. 1, p. 271–281, 2019.

AMARO, H. F. W. S. S. L. Análise da frequência fundamental, jitter, shimmer e intensidade vocal em crianças com transtorno fonológico. *Revista Brasileira de Otorrinolaringologia*, scielo, v. 71, p. 582 – 588, 10 2005. Disponível em: ⟨http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0034-72992005000500007&nrm=iso⟩.

AWAIS, A. et al. Speaker recognition using mel frequency cepstral coefficient and locality sensitive hashing. In: IEEE. *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*. [S.l.], 2018. p. 271–276.

BAUMANN, R. et al. Voice spoofing detection corpus for single and multi-order audio replays. *Computer Speech & Language*, v. 65, p. 101132, 2021.

BENNETT, K. P.; CAMPBELL, C. Support vector machines: hype or hallelujah? *Acm Sigkdd Explorations Newsletter*, Acm New York, NY, USA, v. 2, n. 2, p. 1–13, 2000.

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

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

BOSI, M.; GOLDBERG, R. *Introduction to Digital Audio Coding and Standards*. [S.l.]: Springer US, 2002. (The Springer International Series in Engineering and Computer Science).

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

CONSORTIUM, L. D. *TIMIT Acoustic-Phonetic Continuous Speech Corpus*. 2018. Disponível em: ⟨https://catalog.ldc.upenn.edu/byyear#2018⟩.

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⟩.

DAS, K. A. et al. Modified Gammatone Frequency Cepstral Coefficients to Improve Spoofing Detection. In: . [S.l.]: IEEE, 2016. p. 50–55.

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).

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

GHAZALI, R. et al. *Recent Advances on Soft Computing and Data Mining: Proceedings of the Third International Conference on Soft Computing and Data Mining (SCDM 2018), Johor, Malaysia, February 06-07, 2018*. [S.l.]: Springer International Publishing, 2018. (Advances in Intelligent Systems and Computing).

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.

HANILÇI, C. Linear prediction residual features for automatic speaker verification anti-spoofing. *Multimedia Tools and Applications*, v. 77, n. 13, p. 16099–16111, Jul 2018.

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

IBM AND MICROSOFT. *Multimedia Programming Interface and Data Specifications 1.0*. [S.l.], 1991. Rev. 1.0.

IDIAP RESEARCH INSTITUTE. *AVspoof Database 2015*. 2015. Disponível em: ⟨https://www.idiap.ch/dataset/avspoof⟩.

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

JOHANSSON, M. The hilbert transform. *Mathematics Master's Thesis. Växjö University, Suecia. Disponible en internet: http://w3. msi. vxu. se/exarb/mj_ex. pdf, consultado el*, v. 19, 1999.

JOLLIFFE, I. *Principal Component Analysis*. [S.l.]: Springer New York, 2006. (Springer Series in Statistics).

KAMBLE, M. R.; PATIL, H. A. Novel Energy Separation Based Frequency Modulation Features For Spoofed Speech Classification. In: . [S.l.: s.n.], 2017. p. 326–331.

KAMBLE, M. R.; PATIL, H. A. Novel Variable Length Energy Separation Algorithm using Instantaneous Amplitude Features For Replay Detection. In: Int Speech Commun Assoc. [S.l.], 2018. (Interspeech), p. 646–650.

KINNUNEN, T. e. a. *SAS 2017*. 2017. Disponível em: ⟨https://datashare.is.ed.ac.uk/handle/10283/2778⟩.

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.

KSCHISCHANG, F. R. The hilbert transform. *University of Toronto*, Citeseer, v. 83, p. 277, 2006.

LI, K. S. V. S. E. A. H. Front-end for antispoofing countermeasures in speaker verification: Scattering spectral decomposition. *IEEE Journal of Selected Topics in Signal Processing*, v. 11, n. 4, p. 632–643, June 2017.

PATEL, T.; PATIL, H. Combining evidences from mel cepstral, cochlear filter cepstral and instantaneous frequency features for detection of natural vs. spoofed speech. In: . [S.l.: s.n.], 2015.

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

POOLE, D. *Linear Algebra: A Modern Introduction*. [S.l.]: Cengage Learning, 2014.

RAHMENI, R.; AICHA, A. B.; AYED, Y. B. On the contribution of the voice texture for speech spoofing detection. In: . [S.l.]: IEEE, 2019. p. 501–505.

REDDOTS. *RedDots database*. 2015. Disponível em: ⟨https://sites.google.com/site/thereddotsproject/home⟩.

REN, Y. et al. Replay attack detection based on distortion by loudspeaker for voice authentication. *Multimedia Tools and Applications*, v. 78, n. 7, p. 8383–8396, Apr 2019.

SALOMON, D.; MOTTA, G.; BRYANT, D. *Data Compression: The Complete Reference*. [S.l.]: Springer London, 2007. (Molecular biology intelligence unit).

SAPP, C. S. *WAVE PCM soundfile format*. 2019. Disponível em: ⟨http://soundfile.sapp.org/doc/WaveFormat/⟩.

SARANGI, S.; SAHIDULLAH, M.; SAHA, G. Optimization of data-driven filterbank for automatic speaker verification. *Digital Signal Processing*, v. 104, p. 102795, 2020.

SARANYA, M. S.; PADMANABHAN, R.; MURTHY, H. A. Replay Attack Detection in Speaker Verification Using non-voiced segments and Decision Level Feature Switching. In: IEEE. [S.l.], 2018. (International Conference on Signal Processing and Communications SPCOM), p. 332–336.

SUTHOKUMAR, G. et al. Modulation dynamic features for the detection of replay attacks. In: . [S.l.: s.n.], 2018. p. 691–695.

TODISCO, M.; DELGADO, H.; EVANS, N. Constant q cepstral coefficients: A spoofing countermeasure for automatic speaker verification. *Computer Speech & Language*, v. 45, p. 516 – 535, 2017.

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

WASILEWSKI, F. *Wavelet Properties Browser*: Wavelet haar (haar). 2020. Disponível em: ⟨http://wavelets.pybytes.com/wavelet/haar/⟩.

WICKRAMASINGHE, B. et al. Frequency Domain Linear Prediction Features for Replay Spoofing Attack Detection. In: Int Speech Commun Assoc. [S.l.]: ISCA-INT SPEECH COMMUNICATION ASSOC, 2018. (Interspeech), p. 661–665.

WU, Z. e. a. *SAS 2015*. 2015.

YAMAGISHI, J. e. a. *SAS 2019*. 2019. Disponível em: ⟨https://datashare.is.ed.ac.uk/handle/10283/3336⟩.

YAN, D. et al. Detection of hmm synthesized speech by wavelet logarithmic spectrum. *Automatic Control and Computer Sciences*, v. 53, n. 1, p. 72–79, Jan 2019.

YANG, J.; DAS, R. K. Low frequency frame-wise normalization over constant-q transform for playback speech detection. *Digital Signal Processing*, v. 89, p. 30–39, 2019.

Ye, Y. et al. Detection of replay attack based on normalized constant q cepstral feature. In: . [S.l.: s.n.], 2019. p. 407–411.

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.

Apêndices

APÊNDICE A – Aplicação de um filtro *Wavelet*

Figura 56 – Demostração numérica das transformadas Wavelet e *packet-wavelet*: Por razões didáticas a *Wavelet* Haar foi considerada como tendo os valores $1/2$ e $-1/2$

Packet-wavelet para filtro haar = {0.5, 0.5}

Sinal	32	10	20	38	37	28	38	34	18	24	24	9	23	24	28	34
	h								g							
Nível 01	21	29	32,5	36	21	16,5	23,5	31	11	-9	4,5	2	-3	7,5	-0,5	-3
	h				g				g				h			
Nível 02	25	34,25	18,75	27,25	-4	-1,75	2,25	-3,75	10	1,25	-5,25	1,25	1	3,25	2,25	-1,75
	h		g		g		h		h		g		g		h	
Nível 03	29,62	23	-4,625	-4,25	-1,125	3	-2,875	-0,75	5,625	-2	4,375	-3,25	-1,125	2	2,125	0,25
	h	g	g	h	h	g	g	h	h	g	g	h	h	g	g	h
Nível 04	26,3125	3,3125	-0,1875	-4,4375	0,9375	-2,0625	-1,0625	-1,8125	1,8125	3,8125	3,8125	0,5625	0,4375	-1,5625	0,9375	1,1875

← **resultado do passa-baixas (h)**
← **resultado do passa-altas (g)**

Comparação

Transformada wavelet regular

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

Transformada packet-wavelet

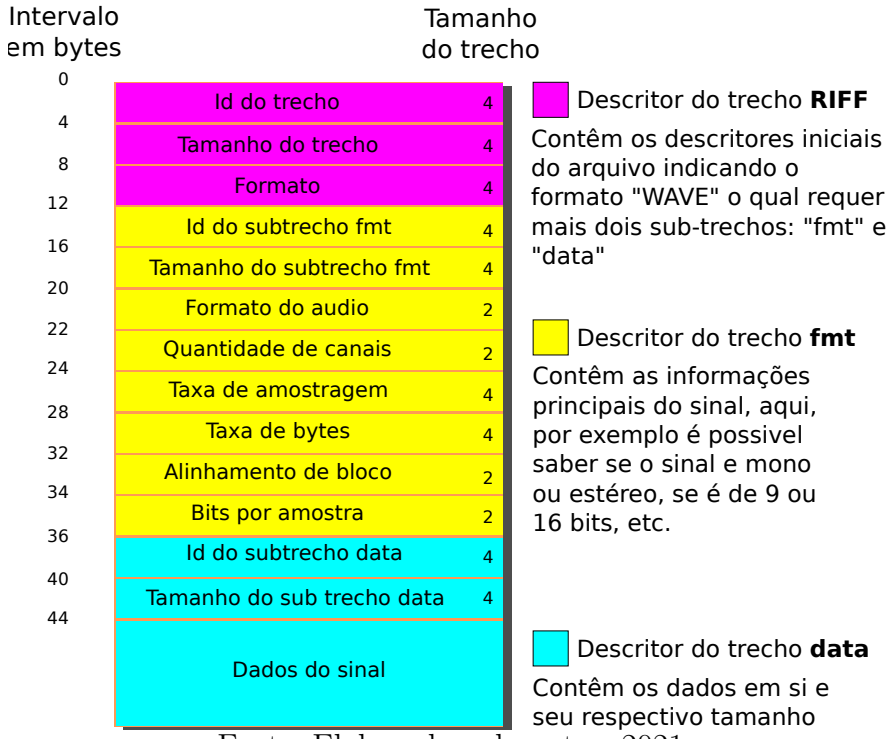
Sinal	32	10	20	38	37	28	38	34	18	24	24	9	23	24	28	34
Nível 01	21	29	32,5	36	21	16,5	23,5	31	11	-9	4,5	2	-3	7,5	-0,5	-3
Nível 02	25	34,25	18,75	27,25	-4	-1,75	2,25	-3,75	10	1,25	-5,25	1,25	1	3,25	2,25	-1,75
Nível 03	29,62	23	-4,625	-4,25	-1,125	3	-2,875	-0,75	5,625	-2	4,375	-3,25	-1,125	2	2,125	0,25
Nível 04	26,3125	3,3125	-0,1875	-4,4375	0,9375	-2,0625	-1,0625	-1,8125	1,8125	3,8125	3,8125	0,5625	0,4375	-1,5625	0,9375	1,1875

Fonte: Elaborado pelo autor, 2021.

APÊNDICE B – Arquivos digitais *wave*

Os arquivos no formato digital *wave*, popularmente utilizados para armazenar áudio e voz digital (SAPP, 2019), assim como ocorre neste trabalho, se estruturam como ilustrado na Figura 57.

Figura 57 – Estrutura do arquivo Wave



Fonte: Elaborado pelo autor, 2021.

A estrutura de interesse se localiza na última parte do arquivo, mais especificamente no bloco “data” ilustrado em azul na Figura. Nele, os dados são organizados como um grande vetor de números, em formato *little endian* (ADIGA, 2007), sendo que cada um deles indica a intensidade do sinal naquele ponto. A descrição pormenorizada do bloco “data”, consultada em (IBM AND MICROSOFT, 1991), foi utilizada neste trabalho para possibilitar a extração dos dados brutos dos sinais utilizados.

APÊNDICE C – Recursos na web

Para realização deste trabalho foram produzidos variados textos (este incluso) e códigos, é recomendada a consulta dos códigos fontes utilizados nos procedimentos como complemento que, se espera, melhore o entendimento dos assuntos discutidos.

Uma atenção especial deve ser dada ao diretório localizado em ***/src/lib*** que contém todas as bibliotecas criadas e referenciadas para resolver os problemas surgidos e também ao ***/src/experiments*** que contém os procedimentos já descritos.

Em ***/soundSamples*** se encontra a base de dados com os áudios originais e tratados.

Alguns *scripts* foram desenvolvidos para facilitar o tratamento dos arquivos de áudio, os mesmos se encontram em ***/src/scripts***.

Por fim a parte escrita pode ser consultada em ***/documentation***.

Todos esses materiais estão sob licença de código aberto (GPLv3) e são livres para uso não comercial acesse https://github.com/ensismoebius/voiceSpoofingDetectionWavelet para mais informações.

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

TERMO DE REPRODUÇÃO XEROGRÁFICA

Autorizo a reprodução xerográfica do presente Trabalho de Conclusão, na íntegra ou em partes, para fins de pesquisa.

São José do Rio Preto, 25 de Agosto de 2021

André Furlan