UNIVERSIDADE FEDERAL DO ABC

ENGENHARIA DE INFORMAÇÃO

LUAN GONÇALVES MIRANDA

AUTENTICAÇÃO DE ASSINATURAS UTILIZANDO TÉCNICAS DE APRENDIZADO DE MÁQUINA

SANTO ANDRÉ

2018

LUAN GONÇALVES MIRANDA

AUTENTICAÇÃO DE ASSINATURAS UTILIZANDO TÉCNICAS DE APRENDIZADO DE MÁQUINA

Relatório apresentado à disciplina "Trabalho de Graduação III" do curso de Engenharia de Informação da Universidade Federal do ABC, como requisito para obtenção do grau de Engenheiro de Informação

Orientador: Prof. Dr. Murilo Bellezoni Loiola

SANTO ANDRÉ

2018

RESUMO

Este trabalho tem o intuito de realizar um estudo e uma análise comparativa de técnicas de aprendizado de máquina (do inglês *Machine Learning*), como redes neurais artificiais, algoritmos bio-inspirados e máquinas de vetores de suporte, por exemplo, para verificação de autenticidade de assinaturas. Para isto, foram utilizadas assinaturas contidas em bancos de dados disponíveis gratuitamente na internet. Estes bancos de dados disponibilizam um conjunto de assinaturas, verdadeiras e falsas, para o teste de sistemas como os que foram implementados neste trabalho.

Palavras-Chaves: Aprendizado de Máquina; Processamento de Sinais; Processamento de Imagens; Redes Neurais; Algoritmos Bio-inspirados; Máquinas de vetores de suporte; Autenticação de assinaturas.

ABSTRACT

This work intends to carry out a study and a comparative analysis of Machine Learning techniques, such as artificial neural networks, bio-inspired algorithms and support vector machines, to verify the authenticity of handwritten signatures. To that end, signatures contained in databases freely available on the internet were used. These databases provide a set of true and false handwritten signatures for testing systems such as those were implemented in this work.

Keywords: Machine Learning, Signal Processing, Image Processing, Neural Networks, Bio-inspired Algorithms, Support Vector Machines; Signature authentication..

SUMÁRIO

1. INTRODUÇÃO

A segurança digital vem sendo um dos tópicos mais importantes nos dias atuais, tendo em vista que grande parte da informação do mundo trafega por meios digitais. Com isso, meios de proteção e autenticação dessas informações se tornaram indispensáveis.

"O que você sabe" ou "o que você é" são as duas abordagens utilizadas para proteção e autenticação de informação em segurança digital. Senhas, frases e perguntas são as formas de segurança que utilizam a abordagem "O que você sabe". Já na abordagem "O que você é", as principais características são as biométricas [1]. Biometria refere-se às características únicas encontradas em cada indivíduo, podendo ser classificadas como fisiológicas ou comportamentais. As mais comuns biometrias fisiológicas incluem as características faciais, impressões digitais, retina ou íris, geometria da mão, etc. As biometrias comportamentais incluem padrões de voz, de caminhada, de assinatura, etc [2].

Ultimamente, as tecnologias que utilizam biometrias comportamentais vêm ganhando bastante espaço na segurança digital e vêm sendo muito estudadas, tendo em vista que esses padrões comportamentais muitas vezes são únicos e não podem ser roubados (como pode ocorrer com as senhas) ou são difíceis de serem copiados (como podem acontecer com cópias de digitais em silicone).

Um dos padrões biométricos comportamentais mais utilizado no mundo é o de assinatura, porém é uma das biometrias comportamentais passível de cópias. A assinatura é uma das formas de justificar ou validar um documento. Sendo assim, a sua fraude é uma das principais formas de perda financeira, tendo em vista que a maioria dos documentos e contratos tratam de acordos envolvendo dinheiro. Com isso, tecnologias e formas de verificar a autenticidade de uma assinatura se tornaram uma necessidade nos dias atuais. Portanto, este trabalho busca realizar um estudo sobre algumas técnicas de verificação de

autenticidade de assinaturas utilizando o aprendizado de máquina como ferramenta.

Abaixo é mostrado um diagrama esquemático das etapas seguidas e a organização deste trabalho:

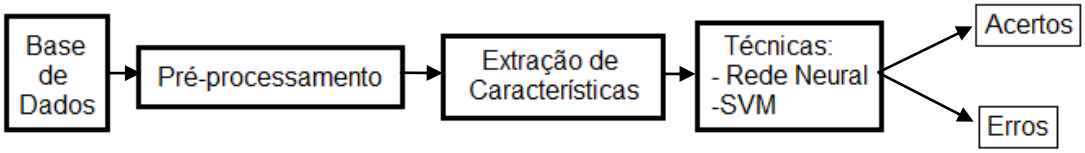


Figura 1: Diagrama das etapas utilizadas para a autenticação de assinaturas.

Inicialmente as assinaturas das bases de dados passaram por algumas etapas de pré-processamento, pois as assinaturas presentes nestas bases eram assinaturas brutas, ou seja, assinaturas que foram colhidas e digitalizadas sem processamento. Em seguida foram extraídas algumas características dessas imagens, para depois serem utilizadas como entrada nos sistemas implementados utilizando as técnicas de aprendizado de máquina (como a Rede Neural, SVM que foram utilizadas neste trabalho). Por fim, foram obtidos os resultados para cada sistema, onde o resultado refere-se a quantidade e acertos e erros cometidos pela rede.

2. PESQUISAS ATUAIS

Atualmente, diversas pesquisas sobre biometria vêm sendo feitas tanto no mundo científico, quanto no mundo corporativo e a autenticação de assinaturas utilizando aprendizado de máquina é um desses temas que vêm sendo estudados.

Alguns trabalhos utilizam técnicas já conhecidas no reconhecimento de padrões, como redes neurais, máquinas de vetores de suporte e algoritmos baseados no sistema imunológico, como é o caso dos artigos [3-8]. Nos artigos [3] e [4] foram utilizadas redes neurais; em [5] e [6], sistemas imunológicos artificiais (AIRS, do inglês *Artificial Immune Recognition Systems*) e em [7] e [8], máquinas de vetores suporte (SVM, do inglês *Support Vector Machines*).

Especificamente, em [3] e [4] foram utilizadas redes neurais do tipo perceptron multicamada (MLP, do inglês *Multilayer Perceptron*). A primeira diferença entre eles, no entanto, é que foi utilizada aquisição *offline* e *online*, respectivamente. Essa é uma diferença importante pelo fato de que na aquisição *online*, onde se utiliza um equipamento específico (*tablet*) para colher as assinaturas, é possível obter diversos parâmetros (como pressão ao escrever, inclinação da caneta, direção, etc.) que facilitam a identificação da assinatura. Já na aquisição *offline*, onde a assinatura é colhida em papel e posteriormente é digitalizada, não se pode utilizar nenhum destes parâmetros disponíveis na aquisição *online*. Nos dois artigos citados acima foi seguida uma mesma sequência de processos: aquisição das assinaturas, pré-processamento das imagens, extração de características, treinamento da rede e, por fim, teste do sistema. Em [3] foi utilizada uma base de dados de assinaturas disponível na internet para os testes do sistema. No pré-processamento ambos fizeram as mesmas etapas, que serão descritas nas seções posteriores deste trabalho. Outra diferença entre [3] e [4] foi a extração das características. No artigo [3] foram utilizados os coeficientes da transformada *wavelet* das imagens como características extraídas, enquanto que em [4], características como centro de massa, transição da esquerda para direita e área normalizada foram utilizadas nessa etapa. No treinamento foi utilizado o algoritmo *back propagation* [9] em ambos os casos.

Nos artigos [5] e [6] foi utilizada a técnica de sistema imunológico artificial para a autenticação de assinaturas. Em [5] é proposta a utilização do sistema imunológico artificial juntamente com máquina de vetores suporte. No AIRS convencional, são realizadas três etapas diferentes: inicialização, treinamento e classificação. Na classificação, depois do treinamento de todos os antígenos, utiliza-se a MC (Célula de Memória, do inglês *Memory Cell*) para a predição das amostras de teste sem considerar qualquer outra informação útil do conjunto de treinamento inicial. Esta MC utilizada é baseada em um treinamento similar ao usado no classificador SVM. Com isso, é possível juntar o AIRS e SVM para um melhor funcionamento do sistema.

Já em [6] é feito o estudo somente utilizando a técnica convencional do AIRS. Para isso, ele utiliza para a geração das características a

transformada Ridgelet [10], que é uma técnica criada para substituir a transformada wavelet no processamento de imagens de duas dimensões. Em seguida, foram feitas as etapas descritas acima para o artigo [5]: inicialização, treinamento e classificação, onde na inicialização a distância Euclidiana dos dados (distância em relação a cada um dos elementos do vetor de características para os outros) é normalizada, sendo dimensionada na faixa de [0,1]. No treinamento, deve ser encontrado o protótipo MC que tem a maior estimulação para gerar um conjunto de clones mutados aleatoriamente e, na classificação, ocorre a decisão dos vizinhos mais próximos.

Nos artigos [7] e [8] foram utilizadas a técnica de máquina de vetor suporte para o desenvolvimento dos estudos. Em [7] foi utilizada a técnica SVM clássica para o desenvolvimento da pesquisa, ou seja, foi utilizado o algoritmo conforme foi descrito por seus criadores. O sistema desenvolvido é dividido em duas etapas: treinamento e reconhecimento das assinaturas dadas. Para isso, foram feitas as etapas de pré-processamento, também utilizadas em [3] e [4], que serão descritas posteriormente.

Foram utilizadas três tipos de características: global, de mascaramento e de grade. As características globais (*Global features*) dão informações sobre casos gerais da forma da assinatura (por exemplo, características que identificam que a imagem é uma assinatura); as de mascaramento (*Mask feature*) dão informações sobre a direção das linhas da assinatura e as de grade (*Grid feature*) dão informações gerais sobre a aparência da assinatura.

Para o treinamento e teste dos sistemas foram utilizadas 1320 assinaturas colhidas de 70 pessoas. No treinamento, 40 voluntários participaram do estudo, sendo que cada um gerou 8 assinaturas genuínas. Além disso, foram utilizadas 4 assinaturas forjadas por pessoa e 30 imitações dessas assinaturas (forjada significa que outra pessoa copiou uma assinatura genuína e imitada significa que a mesma pessoa que fez a assinatura genuína copiou sua própria assinatura[1]).

Após isso, o sistema foi utilizado de duas formas diferentes: para verificação, sendo a verificação a decisão se uma assinatura é verdadeira ou

[1] A assinatura imitada é uma forma de simular uma tentativa de fraude ao sistema, onde a pessoa altera algumas características de sua assinatura com o intuito de induzir o sistema a dizer que não foi ela que realizou alguma ação, por exemplo, assinatura de um contrato.

falsa, e para o reconhecimento de uma assinatura, onde o sistema indica a identidade do dono da assinatura. Para o processo de verificação foram utilizadas 8 assinaturas originais e 8 forjadas, tendo uma taxa de acerto de 98%. Por fim, foi feito o reconhecimento utilizando apenas assinaturas genuínas (isto porque o objetivo era identificar uma pessoa, não sendo possível realizar esta tarefa utilizando assinaturas falsas), proporcionando uma taxa de acerto de 95%.

Em [8], a técnica SVM foi utilizada para o reconhecimento das assinaturas utilizando uma GPU (do inglês *Graphics Processing Unit*, em português Unidade de Processamento Gráfico). O trabalho pode ser dividido em duas partes: extração de características de uma base de dados selecionada e utilização dessas características para verificação de uma imagem dada. Além disso, o método proposto utiliza a função de Kernel Universal (*Universal Kernel Function*). A base de dados utilizada possuía dados de 300 indivíduos, sendo que cada indivíduo possuia 54 assinaturas (24 originais e 30 forjadas). Como características extraídas, foram utilizados diversos parâmetros. Um exemplo é o ângulo do centro de gravidade (*Gravity Center Angle*), onde a imagem é dividida em duas utilizando uma linha vertical em seu centro, cortando-a em duas seções de tamanhos iguais. Depois disso, é calculado o centroide de cada uma dessas duas seções. Por fim, o ângulo do vetor que passa por esses dois centroides é utilizado como característica final extraída. Depois disso, foi implementado o SVM na GPU. Por fim, foram realizados os testes do sistema.

Nos tópicos a seguir, são descritos, além dos objetivos deste trabalho, as etapas necessárias que foram feitas para o objetivo proposto.

3. OBJETIVOS

Este trabalho tem como objetivo realizar um estudo comparativo de técnicas de aprendizado de máquina, como redes neurais, máquina de vetor de suporte e algoritmos bio-inspirados, para realização de autenticação de assinaturas.

4. AQUISIÇÃO DE ASSINATURA (*ONLINE* E *OFFLINE*)

As aquisições das assinaturas podem ser feitas de duas maneiras: *online* ou *offline.* Na aquisição *online*, geralmente, é utilizado um equipamento mais sofisticado, como um tablet, onde diversos parâmetros podem ser obtidos para a autenticação. Inclinação da caneta, velocidade e pressão são algumas das características que são levadas em conta para autenticação utilizando essa abordagem. Já na aquisição *off-line*, a aquisição é feita através da digitalização de um documento onde a assinatura se encontra, se atentando ao fato de que os diversos parâmetros utilizados na autenticação *online* não estão disponíveis para autenticação [3, 4].

No dia a dia, a maior parte das assinaturas é colhida de forma *offline*. Levando isso em consideração, e o fato de que o equipamento para aquisição online não é financeiramente viável para este trabalho, foi utilizada a aquisição *offline* para o desenvolvimento deste trabalho. Além disso, como já existem bancos de dados de assinaturas disponíveis gratuitamente na internet, foi escolhido e utilizado um desses bancos para os treinamentos e testes dos sistemas implementados.

4.1. Base de dados

Nesses bancos de dados, é disponibilizado um conjunto de assinaturas para realização de todas as etapas do processo de reconhecimento e autenticação: treinamento, teste e reconhecimento.

Para realização desde projeto, foram escolhidos dois bancos de dados disponíveis em [11] e [12]. Essas são bases de dados que contém amostras de assinaturas *offline* utilizadas em competições realizadas sobre reconhecimento de padrões. Coletadas sob supervisão de Bryan Found e Doug Rogers, elas foram colhidas entre os anos 2001 e 2006, sendo as imagens digitalizadas com 600 dpi e/ou 300 dpi de resolução [11, 12].

Nesta competição, é utilizado dois bancos de dados, sendo um para treinamento do sistema proposto e outro para teste do mesmo.

O conjunto de treinamento [12] contém assinaturas de duas pessoas, ‘A’ e ‘B’ (são utilizados codinomes para preservar a identidade das pessoas utilizadas na coleta). A pessoa ‘A’ possui 200 assinaturas questionáveis, sendo dessas 200 questionáveis, 76 genuínas, 104 forjadas e 20 imitadas. A pessoa ‘B’ possui 100 assinaturas questionáveis, sendo dessas 100 questionáveis, 3 genuínas, 90 forjadas e 7 imitadas [11, 12].

O conjunto de teste [11] é fornecido pelo *Forensic Expertise Profiling Laboratory (FEPL)* da Universidade La Trobe na Austrália. Este conjunto contém amostras de assinaturas de 3 pessoas ‘A1’, ‘A2’ e ‘A3’. As assinaturas questionáveis são uma mistura de assinaturas genuínas, forjadas e imitadas [11, 12]. Porém, como as bases de dados utilizadas para teste possuem assinaturas de pessoas diferentes e para o objetivo deste projeto é necessária a utilização de assinaturas de uma mesma pessoa, o conjunto de treinamento e o conjunto de teste foram divididos para treinar 4 sistemas diferentes, ou seja, um sistema para cada conjunto de assinaturas, sendo utilizados o conjunto A, A1, A2 e A3. O conjunto B não foi utilizado pelo fato de possuir poucas assinaturas genuínas, sendo em sua maioria assinaturas forjadas. Com isso, foi utilizada a proporção 70/30 dessas bases de dados, ou seja, 70% das imagens para treinamento e 30% para validação e teste do sistema.

Como esses bancos de dados escolhidos contém as imagens originais colhidas, foi necessária a realização das etapas de pré-processamento, citadas a seguir, antes de iniciar o treinamento das técnicas de classificação.

Por simplicidade, os conjuntos de dados foram nomeados como: conj 1 (possui 200 assinaturas), conj 2 (possui 250 assinaturas), conj 3 (possui 100 assinaturas) e por fim conj 4 (possui 100 assinaturas).

5. PROCESSAMENTO DAS IMAGENS

5.1. Pré-processamento

Uma das principais etapas no processo de sistemas de reconhecimento de padrões em assinaturas de maneira *offline* é o pré-processamento das

imagens digitalizadas. Ele se torna necessário pelo fato de que ao digitalizar a assinatura utilizando um *scanner*, perde-se qualidade de imagem, além de ser adicionado ruído. Com o pré-processamento, a qualidade pode ser retomada e o ruído, que atrapalharia nas etapas seguintes do sistema, é removido. Abaixo, podem ser observadas algumas etapas de pré-processamento [3, 7]:

A) Eliminação de fundo:
   Nesta etapa, é retirado o fundo da imagem. Isso é feito para que se tenha apenas o contorno da assinatura a qual se deseja autenticar.

B) Conversão da imagem em escala de cinza (Gray scale):
   Nesta etapa é feita a conversão da imagem RGB[2] (colorida), em uma imagem em escala de cinza. Isso é feito pelo fato de que a cor da caneta utilizada não interfere na autenticação da assinatura.

C) Filtragem de ruído:
   Como dito anteriormente, ao digitalizar uma imagem ela adquire ruído. Com isso, esta etapa é necessária para que o ruído adicionado à imagem seja eliminado.

D) Normalização de espessura:
   Diferente da análise *online*, na *offline* a espessura da assinatura não é uma característica analisável. Com isso, é feita uma normalização da espessura do contorno da assinatura para que seja eliminada a diferença de espessura que a caneta proporciona.

E) Normalização da Imagem:
   Assinaturas escritas por uma mesma pessoa podem diferir de tamanho. Para que seja feita uma análise mais precisa, é necessário que todas as assinaturas apresentem um tamanho padrão. Geralmente é feito o ajuste do comprimento da imagem, ficando todas as imagens com o mesmo comprimento.

[2] Refere-se ao sistema de cores que é composto pelas três cores: Vermelho (*Red*), Verde (*Green*) e Azul (*Blue*).

5.2. Extração de Características

Outra etapa também muito importante no processamento de imagens é a extração de características. Nesta etapa, são extraídas das imagens uma ou mais características que possam ser comparadas entre as diversas imagens utilizadas, possibilitando assim a obtenção de um padrão de comportamento em uma assinatura.

Existem dois tipos de características: características de função (*function features*) e características de parâmetros (*parameter features*). As de função são utilizadas na reorganização de assinaturas *online*. Já as de parâmetro se dividem em outros dois tipos: locais e globais. As globais referem-se às características gerais da imagem e as locais, às características específicas das imagens (pixel a pixel) [3].

Neste trabalho, foram utilizadas características de parâmetros locais devido ao fato de ter sido utilizada uma função que será descrita na seção 7.2. Essa função, assim como na referência [3], utiliza a Transformada Wavelet para decompor a imagem e em seguida extrair alguns parâmetros (características) de cada imagem pré-processada.

6. APRENDIZADO DE MÁQUINA

O aprendizado de máquina pode ser descrito como o conjunto de técnicas de inteligência computacional utilizado para análise de dados de maneira automática. Elas recebem este nome pelo fato de que os algoritmos aprendem iterativamente através dos dados, possibilitando que computadores gerem informações que seres humanos não teriam [13].

O aprendizado de máquina é bastante utilizado também em reconhecimentode padrões. Abaixo, serão apresentadas algumas técnicas que foram utilizadas para o objetivo deste projeto.

6.1. Redes Neurais Artificiais (*Artificial Neural Network*)

Redes Neurais Artificiais são técnicas de aprendizado de máquina inspiradas na função biológica (algoritmos bio-inspirados) do cérebro humano, onde bilhões de neurônios interconectados processam informação de forma paralela [9]. Esta técnica é bastante empregada em aplicações de reconhecimento de padrões.

Comparando o cérebro a um computador, pode-se dizer que o cérebro é um computador altamente complexo, não-linear e paralelo. O cérebro realiza instintivamente reconhecimento perceptivo do ambiente a nossa volta de maneira instantânea. Reconhecer um ambiente, rosto de uma pessoa, cheiros, etc. são algumas das tarefas realizadas rotineiramente pelo nosso cérebro. Isso é possível devido ao que chamamos de “experiência”. Ao viver e presenciar algo, a experiência e, consequentemente, a estrutura de reconhecimento presente em nosso cérebro, vai se acumulando e se adaptando com o passar do tempo.

Como dito anteriormente, uma rede neural é inspirada nesta função do cérebro, ou seja, ela é uma máquina projetada para modelar a maneira como o cérebro realiza uma função, no nosso caso o reconhecimento de padrões, de maneira paralela.

Na referência [9], pode-se encontrar a seguinte definição de redes neurais:

“*Uma rede neural é um processador maciçamente distribuído constituído de unidades de processamento simples, que têm a propensão natural para armazenar conhecimento experimental e torná-lo disponível para o uso. Ela se assemelha ao cérebro em dois aspectos:*

1. *O conhecimento é adquirido pela rede a partir de seu ambiente através de um processo de aprendizagem.*
2. *Forças de conexão entre neurônios, conhecidas com pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido.*”

O cérebro é composto por aproximadamente 86 bilhões de neurônios [14], sendo estes neurônios conectados uns aos outros pelas sinapses. O neurônio são as células presentes no sistema nervoso e apresentam como

principal função, conduzir os impulsos nervosos. Já as sinapses, como já dito, são as conexões entre estes neurônios. Essas conexões iniciam-se (no começo da vida dos indivíduos) sem uma organização e com uma grande densidade de sinapses, sendo as experiências do indivíduo (ligado à ideia de treinamento) a responsável por organizar essas conexões [15].

As redes neurais artificiais são inspiradas nestas ideias de conexões de neurônios para realizar o reconhecimento de padrões utilizando um computador. A figura baixo ilustra uma rede neural artificial multicamada (do inglês *Multiple Layer Perceptron* [MLP], ou em português Perceptron de Múltiplas Camadas):

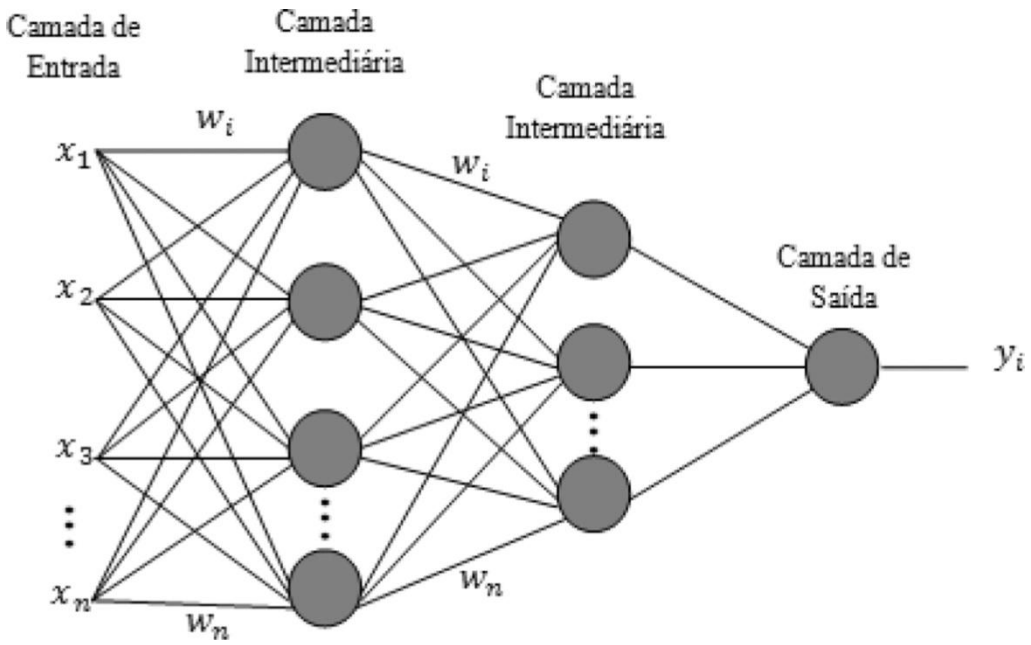


Figura 2: Rede neural artificial (MLP) [16].

A estrutura acima representa uma MLP com duas camadas intermediárias e uma de saída. Camadas intermediárias são onde se encontram os neurônios e suas conexões representam as sinapses destes neurônios. Acima das conexões são representados os pesos sinápticos, sendo esses pesos ajustados no treinamento da rede. O treinamento da rede pode ser de forma supervisionada (quando é dito para a rede a saída desejada para um conjunto de dados no treinamento) ou não-supervisionada (quando não é dito para a rede qual é a saída desejada e ela se auto organiza de forma a

obter os pesos). Neste trabalho foi utilizada a abordagem supervisionada, mais especificamente o algoritmo *backpropagation* [9].

No *backpropagation* é realizada duas etapas de processamento: o processamento direto e o processamento reverso. O objetivo destas etapas é realizar o ajuste dos pesos sinápticos que representam o conhecimento adquirido pela rede [9].

No processamento direto são inseridos na entrada da rede os dados de treinamento obtendo-se a resposta da rede a esses dados. Além disso, é inserido qual o resultado esperado para aquele conjunto de dados. Com o resultado da rede e o resultado esperado, obtém-se um erro associado a esta operação, que define se os pesos sinápticos estão apropriados para o reconhecimento ou não [9].

Com isso, é realizado o processamento reverso. Neste processamento o erro obtido na etapa anterior é retropropagado através da rede, ajustando-se os pesos sinápticos utilizando uma regra de adaptação definida. Este ajuste está ligado a ideia de aprendizado que a rede obteve utilizando os dados de treinamento, tendo em vista que os pesos são inicializados aleatoriamente, assim como ocorre com as redes neurais naturais [9].

Em cada iteração é utilizado um dado de entrada para treinamento. Quando se é utilizado todo o conjunto de dados diz-se que completou-se uma época de treinamento. Pode-se executar o algoritmo para quantas épocas desejar-se ou até atingir uma estabilidade nos pesos (obteve-se a separação entre as classes) [9].

A MLP e o algoritmo *backpropagation* foram os algoritmos utilizados neste trabalho para implementação da rede neural.

6.2. Máquina de Vetor Suporte (*Support Vector Machine*)

Outra técnica de aprendizado de máquina é a chamada Máquina de Vetor Suporte (*SVM*). Diferentemente das técnicas citadas acima, o SVM não é inspirado em nenhuma função biológica e sim na teoria estatística. Ela utiliza a

teoria de aprendizado estatístico para realizar a classificação das amostras de entrada.

Está técnica foi criada por Cortes e Vapnik [17] para classificação binária. A ideia conceitual por trás do *SVM* é a seguinte: Os vetores de entrada são mapeados não-linearmente em um espaço de características de maior dimensão [17], onde as classes associadas ao dados de entrada podem ser separadas (linearmente utilizando um hiperplano ou não linearmente onde é utilizado uma função de kernel para separar as classes). A abordagem do *SVM* pode ser descrita da seguinte maneira [18]:

- Separação de Classe:

  Esta etapa busca encontrar o hiperplano ideal (para classes linearmente separáveis) ou uma função de kernel (para classes não linearmente separáveis) que separa duas classes, maximizando a margem entre os pontos mais próximo dentro da mesma classe. Os pontos que se encontram no limite são os chamados vetores suporte (do inglês, *Support Vector*) e o plano médio da margem é o hiperplano ideal.

- Classes Sobrepostas:

  Os pontos de dados que estão do “lado errado” do hiperplano (ou kernel) são ponderados para reduzir sua influência.

- Não linearidade:

  Quando não é possível encontrar um hiperplano linear, geralmente, os pontos de dados são projetados em um espaço de maior dimensão. Com essa projeção, os dados ficam efetivamente lineares, e isto é feito utilizando as técnicas de kernel, onde uma função mais elaborada é utilizada para realizar a separação das classes..

- Solução do Problema:

Todas as etapas podem ser modeladas como um problema de otimização quadrática, resolvidas assim por técnicas conhecidas.

Entendida todas as etapas necessárias para realizar a autenticação de assinaturas, os próximos tópicos demonstram com foi realiza cada uma dessas etapas utilizando o *software Matlab*.

7. DESENVOLVIMENTO

7.1. Pré-processamento

Seguindo as etapas do pré-processamento descritas na seção 5.1, abaixo se encontram os códigos utilizados para cada etapa de pré-processamento. Além disso, na imagem abaixo é mostrado um exemplo de uma imagem original do banco de dados utilizado, e em cada etapa é mostrado a imagem resultante do pré-processamento realizado:

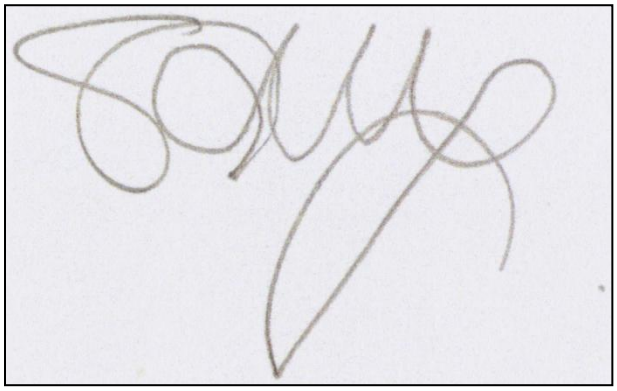


Figura 3: Exemplo de uma imagem original de uma assinatura utilizada neste trabalho

A) Eliminação do Fundo:

```
%%%%%%%%%%%%%%%%%Eliminação de Fundo%%%%%%%%%%%%%%%%%%%%

%Converte imagem RGB para HSV (hue[matiz],saturation[saturação],value[valor])
I = rgb2hsv(I);
%Extraí componentes individual h (matiz), s (saturação) e v (valor)
h = I(:,:, 1);
s = I(:,:, 2);
v = I(:,:, 3);
%Limiar para encontrar cores vivídas
mask = v > 0.7;
%Faz a imagem ficar branca nas áreas da máscara
h(mask) = 0;
```

```
s(mask) = 0;
v(mask) = 1;
%Converte novamente para RGB
I = cat(3, h, s, v);
I = hsv2rgb(I);
```

O código acima utiliza a transformação do padrão de cores RGB para HSV para realizar a remoção do fundo [19]. O padrão de cores HSV é composto por três parâmetros (camadas) (sendo estes parâmetros que nomeiam o padrão):

- Hue (matiz: tonalidade): Responsável pela coloração.
- Saturation (saturação): Refere-se a tonalidade de cinza da imagem.
- Value (valor: brilho): Define o brilho da cor.

Com a extração destes parâmetros, é aplicada uma máscara para deixar o Value (brilho) da imagem branco (ou seja, o brilho máximo). Com isso, todo o fundo ao redor da assinatura ficará branco, ou ainda, será removido.

Por fim, foi feita a concatenação desses parâmetros em uma única imagem e depois esta imagem foi convertida novamente para RGB. A imagem resultante deste processo é mostrada abaixo:

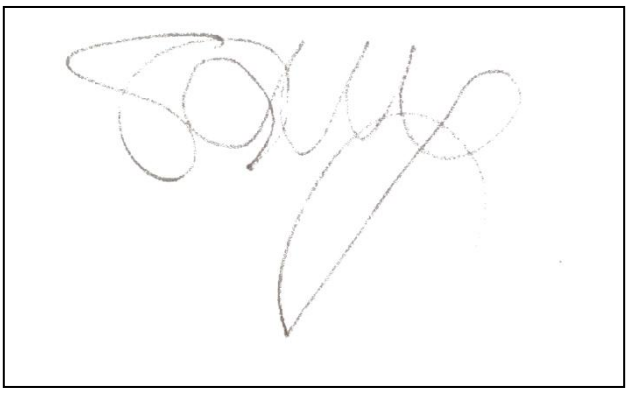


Figura 4: Imagem resultante da eliminação do fundo.

Como pode ser visto, o fundo da imagem foi retirado ficando somente o contorno da assinatura e alguns pontos de ruído.

B) Conversão da imagem em escala de cinza (Gray scale):

```
%%%%%%%%%%%%%%%%%%Converter para Escala de cinza%%%%%%%%%%%%%

I = rgb2gray(I);
```

Para a conversão para da imagem de RGB para escala de cinza foi utilizada a função acima “RGB2GRAY” do Matlab. A imagem resultante deste processo é mostrada abaixo:

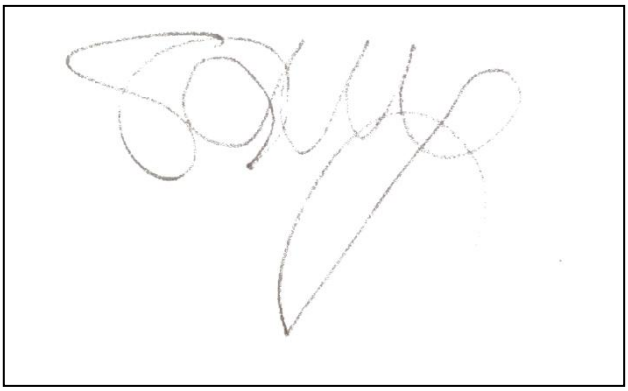


Figura 5: Imagem resultante da conversão para escala de cinza.

Como a imagem anterior já estava em tons de cinza, a imagem permaneceu igual.

C) Filtragem de ruído:

```
%%%%%%%%%%%%%%%%%%Filtragem de ruído (Filtro Mediana)%%%%%%%%%%%%%%

    I = medfilt2(I);
    I = imadjust(I, [0 1], [0 1], 3);
```

Para a filtragem do ruído, foram utilizados dois filtros: um filtro de mediana e um de ajuste de intensidade. O filtro de mediana faz uma filtragem mediana da imagem em duas dimensões e os atualiza conforme o resultado, removendo assim o ruído do fundo e da assinatura em si. Com a remoção do ruído da imagem, os traços da assinatura, que também têm ruído, podem ficar muito claros. Com isso, o filtro de ajuste de intensidade faz com que estes traços se intensifiquem. A imagem resultante deste processo é mostrada abaixo:

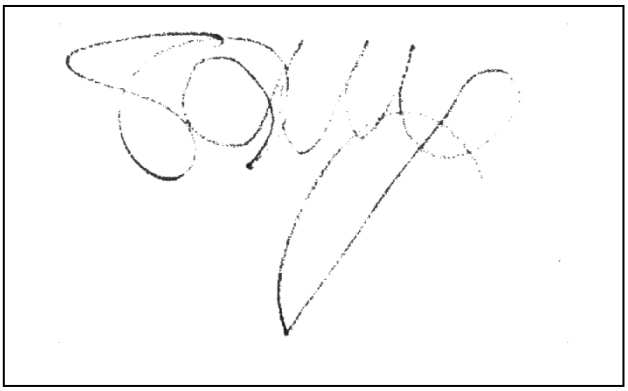


Figura 6: Imagem resultante da filtragem do ruído.

Na imagem pode-se observar que o ruído foi retirado e o contorno da assinatura foi intensificado

D) Normalização de espessura:

```
%%%%%%%%%%%%%%%Normalização da Espessura (Esqueletização)%%%%%%%%%%%

I = imcomplement(I);
I = bwmorph(I,'skel',Inf);
```

O processo de normalização da espessura utiliza uma técnica chamada esqueletização [20]. Na esqueletização, é feita a detecção do objeto (no caso, a assinatura) na imagem e são retirados pixels das bordas deste objeto com o objetivo de, no final do processo, ter apenas uma única linha de pixels que o compõem. O Matlab possui uma função para isto, porém a imagem necessita estar negativada, ou seja, com as cores invertidas. Por este motivo, primeiramente foi feito o negativo da imagem e, posteriormente, sua esqueletização. A imagem resultante deste processo é mostrada abaixo:

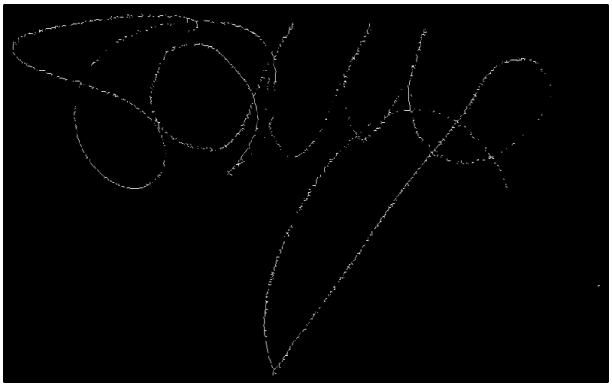


Figura 7: Imagem resultante da normalização da espessura.

Observar-se que o contorno da assinatura ficou com uma única linha de pixels.

E) Normalização da Imagem:

```
%%%%%%%%%%%%%%%%%%%%Normalização da Imagem%%%%%%%%%%%%%%%%%%

%%Detecta Fronteira e Corte da imagem
I = imcomplement(I);
imInvert = ~I;
s = regionprops(imInvert, 'BoundingBox');
z = cat(1, s.BoundingBox);
w = [z(3:end-3,1) z(3:end-3,2) z(3:end-3,1)+z(3:end-3,3) z(3:end-3,2)+z(3:end-3,4)];
xi = min(w(:,1));
yi = min(w(:,2));
xf = max(w(:,3));
yf = max(w(:,4));
s(1).BoundingBox = [xi yi xf-xi yf-yi];
rectangle('Position', s(1).BoundingBox);
imCrop = imcrop(imInvert, s(1).BoundingBox);

%%Normalização da Imagem
I = imresize(imCrop,[NaN 2000]);
```

Para normalizar a imagem, primeiramente é necessário cortá-la, deixando somente a área de interesse (assinatura). Para cortá-la, é feita a inversão da imagem para o fundo ficar preto e o objeto branco. Depois é feita a detecção das fronteiras do objeto utilizando as funções *regionprops* e *boundingbox* [21]. Tendo em vista que com etapas anteriores a assinatura pode não ficar contínua, e essas funções necessitam de continuidade para abranger todo o objeto, foi necessário fazer algumas manipulações dos resultados para obtenção do retângulo que abrange toda a assinatura. Com o retângulo encontrado, é utilizada

a função “*imcrop*” para recortar a imagem somente com o objeto desejado. Depois disso, foi feita a normalização do comprimento da assinatura com o valor desejado e altura proporcional às mudanças. A imagem resultante deste processo é mostrada abaixo:

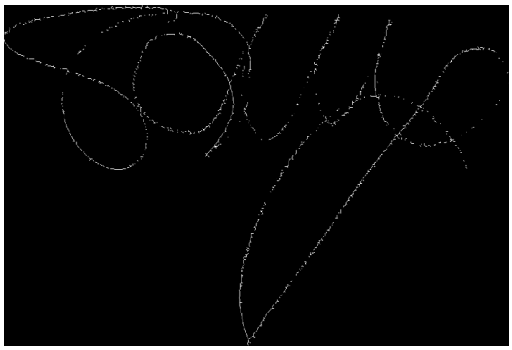


Figura 8: Imagem resultante da normalização da imagem.

Observar-se que houve um corte da imagem em um retângulo que envolve somente o contorno da assinatura, e ainda uma diminuição do tamanho desta.

Ao realizar a normalização do comprimento de um conjunto de imagens, algumas dessas imagens podem perder pontos da assinatura que seriam relevantes para o treinamento e autenticação da mesma. Por isso, foi testado o sistema tanto com a normalização da imagem, quanto com seus tamanhos originais.

7.2. Extração de Características

Como dito anteriormente, a extração de características é uma etapa muito importante do aprendizado de máquina, isso porque ao extrair características de uma imagem, diminui-se a dimensionalidade dos dados que serão utilizados no aprendizado. Isto ocorre pelo fato de não ser viável utilizar a imagem bruta (pré-processada), por ter grandes dimensões, e considerando uma grande quantidade de imagens, torna-se inviável realizar o treinamento com elas.

Para a extração de características, utilizou-se uma função, disponível em [22], que consegue extrair a energia, variância,

comprimento da forma de onda (característica utilizada em imagens médicas, tendo como resultado o comprimento de uma dada forma de onda, sendo essa forma de onda a representação de um estímulo elétrico corporal, e o comprimento desta forma de onda a duração deste estímulo) e a entropia do sinal de entrada (não implementado na versão atual desta função). Para isso, esta função utiliza a decomposição multisinal da transformada wavelet[3]. A função tem quatro argumentos de entrada: o sinal, tamanho da janela (tamanho do bloco de dados que será processado), passo da janela (passo que este bloco irá se deslocar para processar todo o vetor de dados) e a frequência de amostragem (não utilizada na implementação atual). Os valores de janela, passo e frequência utilizados foram 50000, 50000 e 32, respectivamente. Esses valores foram obtidos através de testes e foram utilizados, pois foram os valores que apresentaram tempo razoável para o processamento das imagens.

Esta função realiza a decomposição da imagem utilizando a transformada wavelet em 10 níveis de decomposição. Utilizando a imagem original e as 10 imagens resultantes da decomposição, é possível gerar 11 características (imagem original mais 10 níveis da wavelet). Como estamos extraindo 4 características, ao final do processo teremos 44 características (11*4) extraídas. Além disso, a função utiliza a wavelet 1D, portanto, a imagem foi transformada em um vetor linha para utilização desta função. Dependendo do tamanho deste vetor, a matriz de características gerada terá x linhas (sendo x um número inteiro maior igual a 1, como já dito, dependente do tamanho deste vetor) por 44 colunas, sendo este número x relacionado com a quantidade de janelas utilizadas. Como para a utilização das técnicas de aprendizado de máquina necessitamos de uma matriz com apenas uma única linha (valores das características), foram realizadas duas abordagens para obtenção de um vetor linha através da matriz de características (x linhas por 44 colunas): a primeira foi realizar a soma de todas as linhas e a

[3] A transformada wavelet multisinal é a transformada wavelet de um sinal de uma dimensão, sendo este sinal composto por diversos sinais (de uma Matriz N x M, sendo N e M números inteiros maiores ou iguais a 1) organizados em um vetor linha (ou coluna).

segunda foi realizar a média dos valores das linhas. Ao final do processo, tanto para a soma quanto para a média, obteve-se um vetor 1 X 44, onde as colunas de 1 a 11 representam a energia, 12 a 22, a variância, 23 a 33, o comprimento da forma de onda e de 34 a 44, a entropia do sinal. Como já dito anteriormente, a entropia do sinal não foi implementada nesta versão atual. Com isso, em algumas das 11 características da entropia não conseguimos valores numérico, por isso essas 11 características foram ignoradas no treinamento, sendo assim utilizadas 33 características.

Abaixo, pode-se observar o código utilizado para realizar a extração de características das imagens:

```
%%%%%%%%%%%%%%%%Extração de Características%%%%%%%%%%%%%

%Abre a imagem
X = imread('1p.png');
%Transforma  valores p/ double
x = im2double(X);
%Encontra comprimento e largura
[l,c]=size(x);
%Transforma matriz em vetor coluna
M = reshape(x,l*c,1);
%Seta valor da janela, passo e frequência respectivamente
winsize = 50000;
wininc = 50000;
SF = 32;
%Chama a função
feat = getmswtfeat(M,winsize,wininc,SF);
```

Primeiramente, a imagem foi carregada e depois transformada para o tipo *Double*, pois seus valores estavam definidos como *Logical*, impossibilitando a realização das operações necessárias para extração de características. Em seguida, foi utilizada a função "*size*" para encontrar o comprimento e largura da imagem, valores necessários para realização da transformação, logo em seguida, da matriz em um vetor linha. Depois, foram definidos os parâmetros necessários para a função, que são o valor da janela, passo de processamento e frequência. Por fim, foi feita a chamada da função, passando como parâmetros o vetor obtido, janela, passo e a frequência.

Os vetores de características de todas as imagens foram salvos em um único arquivo texto, sendo os dados deste arquivo organizados em uma matriz de w linhas (sendo w a quantidade de imagens utilizadas) por 44 colunas (44 características extraídas).

A função utilizada para extração das características pode ser encontrada no Apêndice.

7.3. Implementação das Técnicas de aprendizado de máquina

7.3.1. Rede Neural

Para a implementação da rede neural artificial do tipo MLP foi utilizada a *toolbox Neural Pattern Recognition* do Matlab (acessada através do comando nprtool). Nesta *toolbox,* é necessário definir as entradas e saídas desejadas, a proporção de amostras de treinamento, validação e teste, a quantidade de neurônios nas camadas escondidas e o tipo de treinamento.

Inicialmente, deve-se escolher qual é o conjunto de dados de entrada, juntamente com os resultados de saída desejados e a ordenação destes dados (em coluna ou em linha). Os dados utilizados neste projeto estão ordenados em uma matriz linha, ou seja, cada linha corresponde a uma entrada.

Depois é dito qual a proporção dos dados utilizados em cada etapa. Como dito anteriormente, os dados foram divididos em uma proporção 70/30, neste caso, 70/15/15, 70% para treinamento, 15% para validação e 15% para testes. A validação e o teste podem serem considerados iguais, com a resalva que a validação tem interferência no treinamento, sendo esta utilizada como critério de parada do treinamento.

Para terminar os ajustes da rede, o último parâmetro a ser ajustado é a quantidade de neurônios na camada escondida. Para

teste da rede neural, foram variadas as quantidades de neurônios na camada escondida, considerando 10, 30 e 90 neurônios.

Por fim, é realizado o treinamento da rede. Com isso, são apresentados os resultados obtidos pela rede treinada. O algoritmo utilizado para o treinamento é o *Backpropagation* e a função de ativação dos neurônios da camada escondida é do tipo sigmoidal.

7.3.2. SVM

Para implementação da SVM foi utilizada a *toolbox* do Matlab. Nesta *toolbox*, utiliza-se a função *svmtrain* para realizar o treinamento. Esta função possui diferentes parâmetros, possibilitando que sejam realizadas diversas combinações e ajustes do sistema. Um dos principais parâmetros é o *kernel_function*, que permite especificar a função usada para mapear os dados de treinamento dentro do espaço do kernel [23]. As seguintes funções podem ser especificadas no *kernel_function*:

- '*linear*': *kernel* linear, que significa produto escalar.
- '*quadratic*': *kernel* quadrático.
- '*polynomial*': *kernel* polinomial (ordem padrão 3). Através do parâmetro '*polyorder*', pode-se especificar outras ordens do polinômio.
- '*rbf*': Função de *kernel* de Base radial gaussiana, com um fator de escala padrão, sigma, igual a 1. Através do parâmetro '*rbf_sigma*', pode-se especificar outros valores de sigma.
- '*mlp*': *kernel* Perceptron Multicamada, com escala padrão [1-1]. Através do parâmetro '*mlp_params*', pode-se especificar outros valores de escalas

Depois de realizado o treinamento, utiliza-se a função *svmclassify* para realizar a validação do modelo treinado.

Abaixo é mostrado o código utilizado para realização do treinamento e da validação da SVM:

```
% Abre o arquivo com as características extraídas
x = csvread('feature.txt');
% Abre o arquivo com as saídas desejadas (sendo 0 para assinatura falsa, e 1
para verdadeira)
desejado = textread('desejado.txt');

%Definições
nSamples = 200;%Regula número de amostras
N_treino = round(.7*nSamples); % 70% das amostras para treinamento

% Realiza o treinamento da SVM utilizando 140 amostras, 33 características
(Energia, variância e comprimento da forma de onda) e kernel RBF com sigma 2
Train = svmtrain(x(1:N_treino, 1:33),desejado(1:N_treino), 'Showplot', true,
'kernel_function', 'rbf','rbf_sigma', 2);

% Valida o modelo treinado com 60 amostras
Group = svmclassify(Train,x(N_treino+1:end, 1:33),'Showplot',true);

% Mostra a quantidade de erros cometidos na validação
Erros = sum(xor(desejado(N_treino+1:end),Group))
```

No código acima, primeiramente é carregado o arquivo txt contendo as características extraídas e o txt com os valores desejados[4] do conjunto de imagens utilizado. Em seguida são definidos os valores amostras utilizadas no sistema (no caso acima são 200 imagens utilizadas) e a quantidade destas amostras utilizadas para o treinamento da SVM (140 imagens, ou 70% das amostras, utilizadas para treinamento). Depois é realizado o treinamento da SVM através do comando *svmtrain* utilizando as amostras reservadas para treinamento juntamente com os valores desejados. E para esse exemplo foi utilizado o kernel “rbf” com parâmetro “sigma” igual a 2. Por fim, é feito o teste e validação da SVM através do comando *svmclassify* utilizando as amostras das imagens restantes, e o cálculo da quantidade de erros que a SVM cometeu.

[4] Como foi utilizado o treinamento supervisionado, é necessário fornecer a SVM um conjunto de dados que diz se aquele conjunto de características refere-se a uma assinatura original ou não. Neste caso, foi utilizado ‘1’ para assinatura genuína e ‘0’ para falsa.

8. RESULTADOS

Como dito na sessão 4.1, foram utilizados 4 conjuntos de dados, que foram testados utilizando, na extração de características, a soma dos coeficientes e a média dos coeficientes. Os resultados são mostrados nas sessões seguintes.

8.1. Rede Neural

Para a apresentação dos resultados da rede neural foram utilizadas matrizes de confusão. Nestas matrizes são mostradas as quatro possibilidades de ocorrência e resultado na rede: assinatura é falsa e a rede reconhece como falsa (verdadeiro negativo), assinatura é verdadeira e a rede reconhece como falsa (falso negativo), assinatura é falsa e a rede reconhece como verdadeira (falso positivo) e a assinatura é verdadeira e a rede reconhece como verdadeira (verdadeiro positivo). Esses resultados são mostrados da seguinte forma:

Tabela 1: Demonstração da matriz de confusão

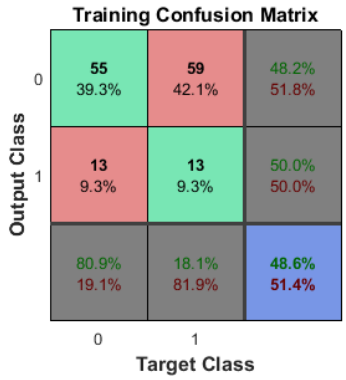
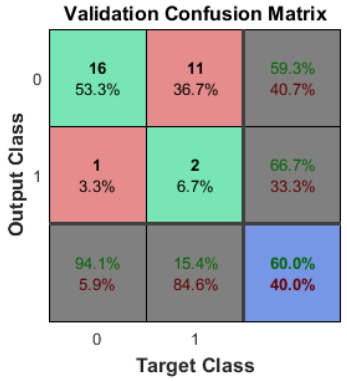
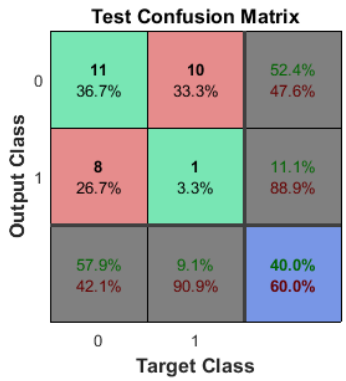
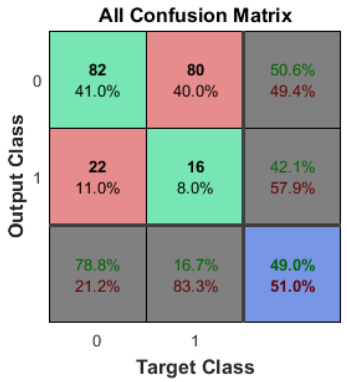
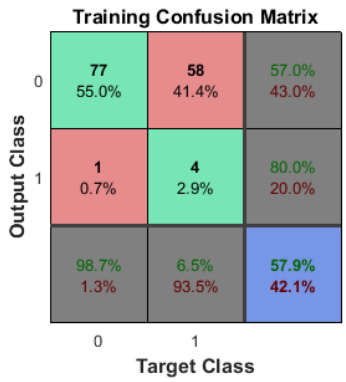
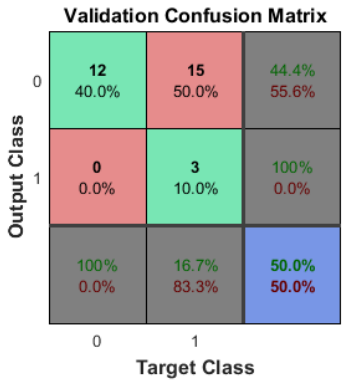
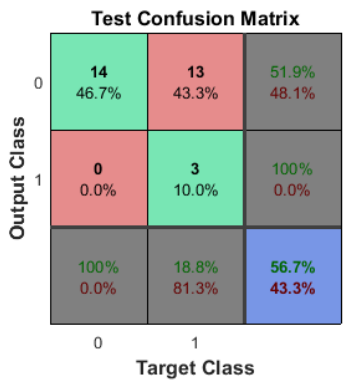
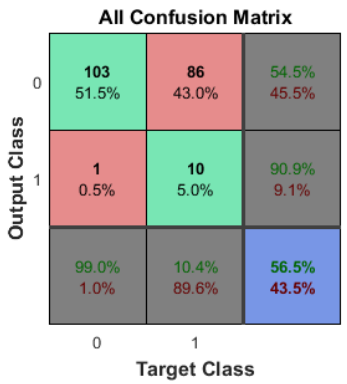
Saída da Rede	0	Verdadeiro Negativo	Falso Negativo
	1	Falso Positivo	Verdadeiro Positivo
		0	1
		Saída Desejada	

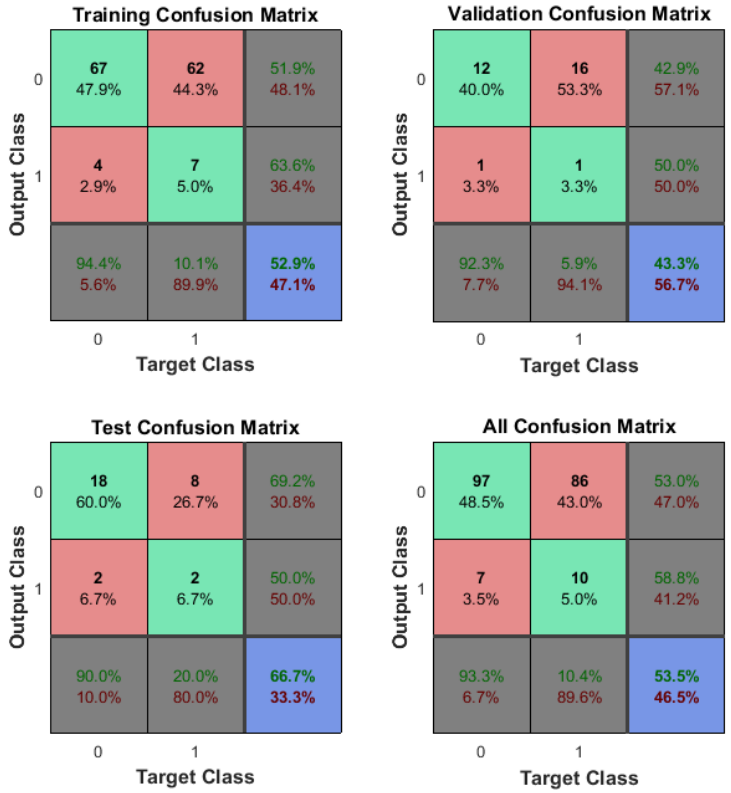
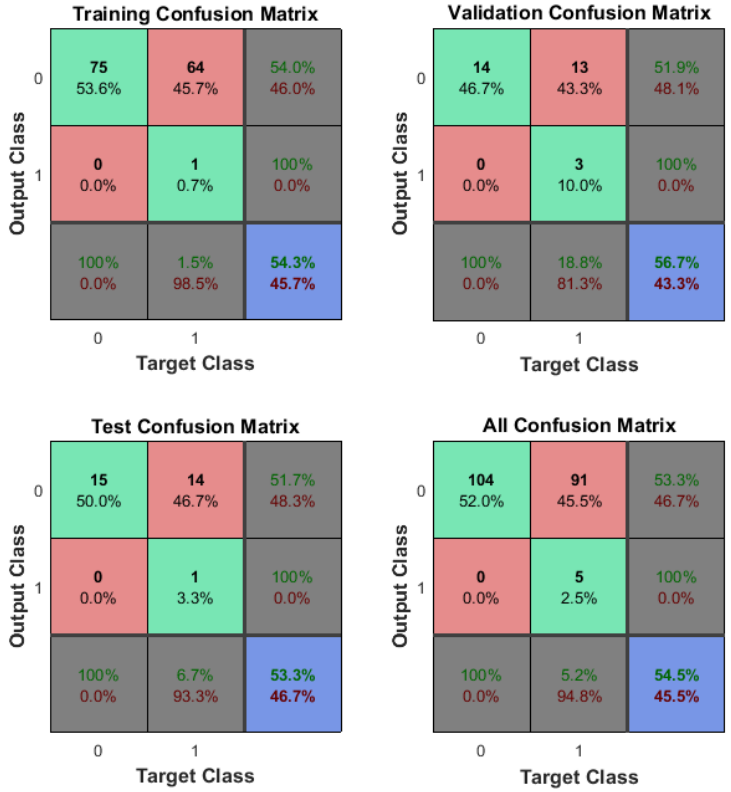
Para cada etapa do processo de reconhecimento da rede neural é apresentada uma matriz de confusão. As seguintes etapas são mostradas: treinamento, validação[5], teste, total (soma dos valores das outras etapas).

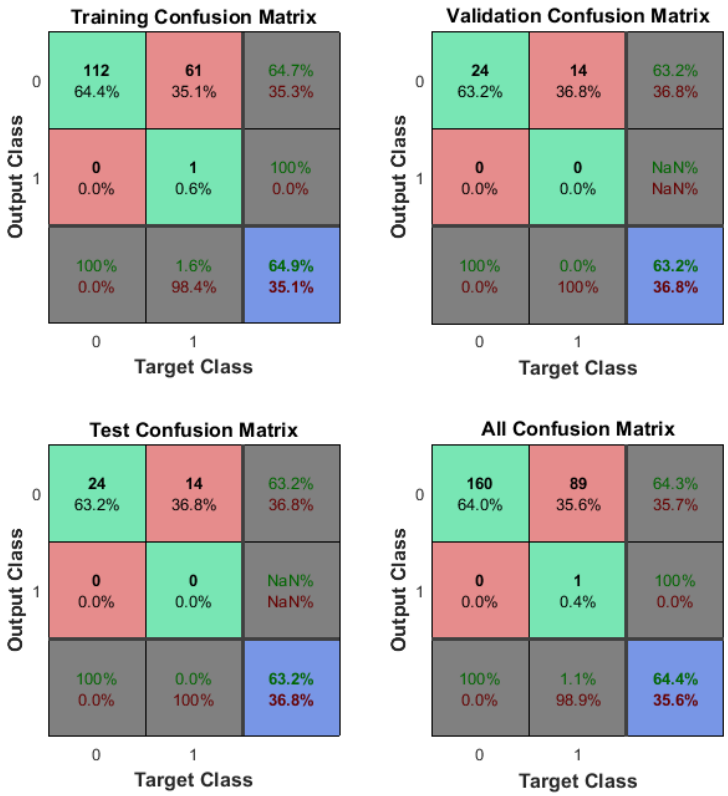
Variando a quantidade de neurônios na camada escondida, as seguintes porcentagens de acertos e erros obtidas para cada uma das bases de dados são mostradas abaixo através das matrizes de confusão. As quantidades de neurônios escolhidos foram: 10, 30 e 90.

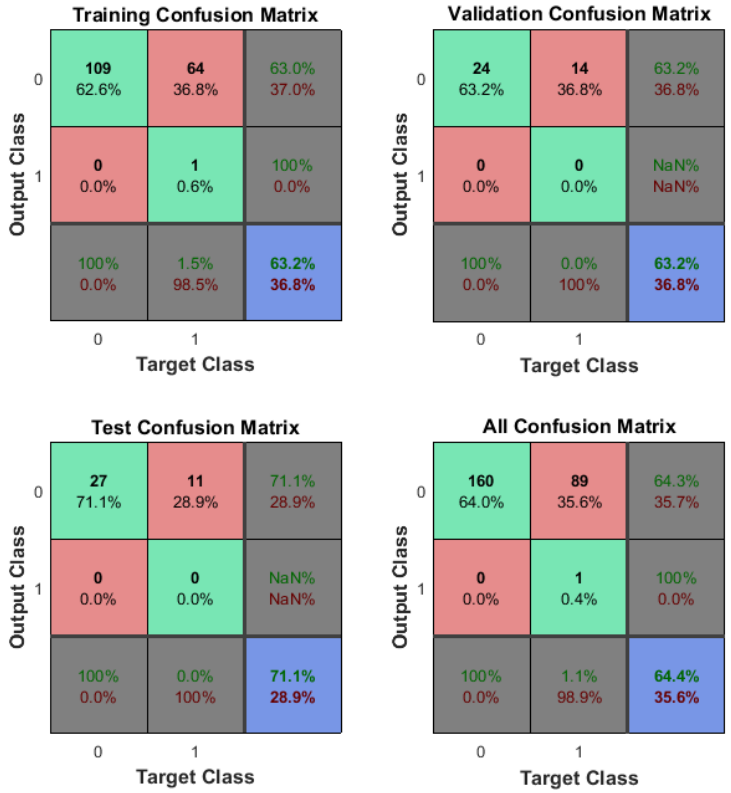
[5] A validação é uma etapa de teste onde ela é utilizada para interromper o treinamento quando a generalização da rede para de melhorar

Conj. 1		
10 NEURÔNIOS	SOMA	
	MÉDIA	

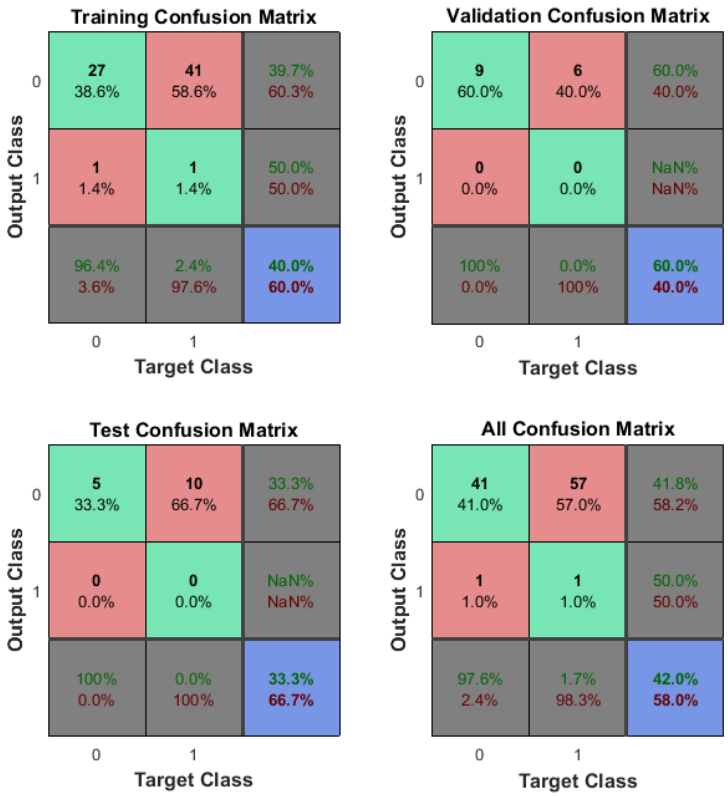
30 NEURÔNIOS	SOMA	   
	MÉDIA	   

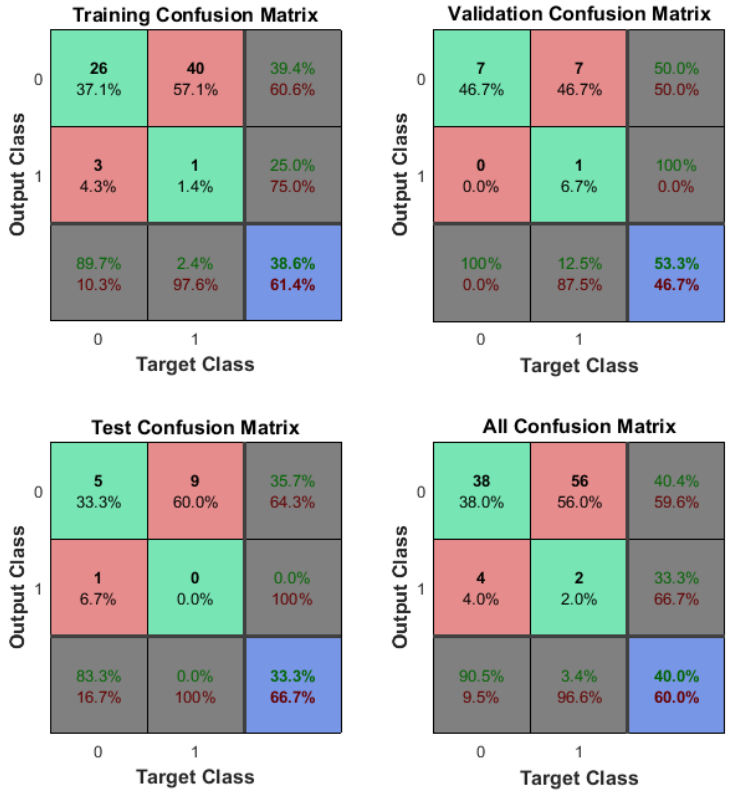
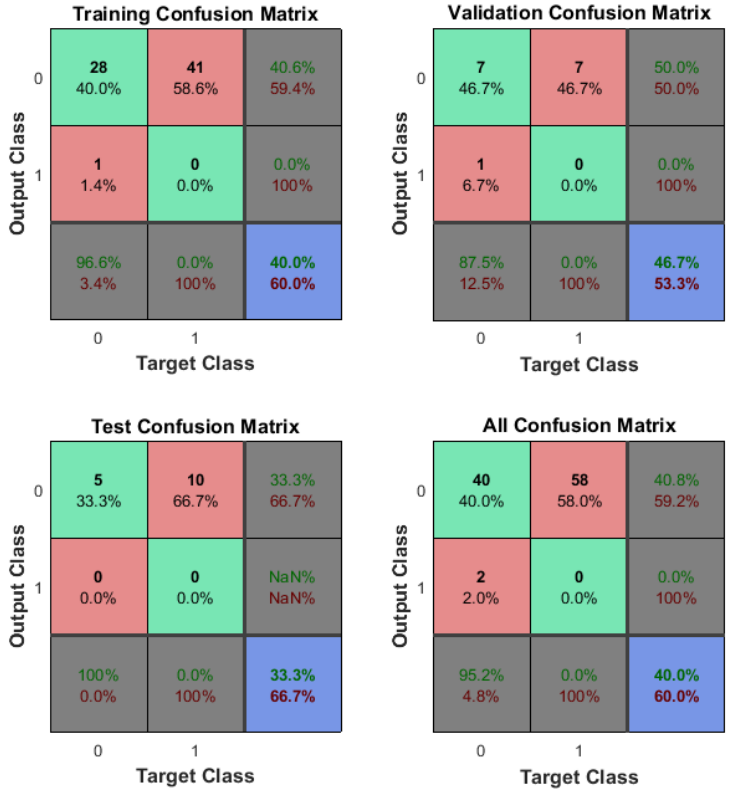
90 NEURÔNIOS	SOMA	
	MÉDIA	

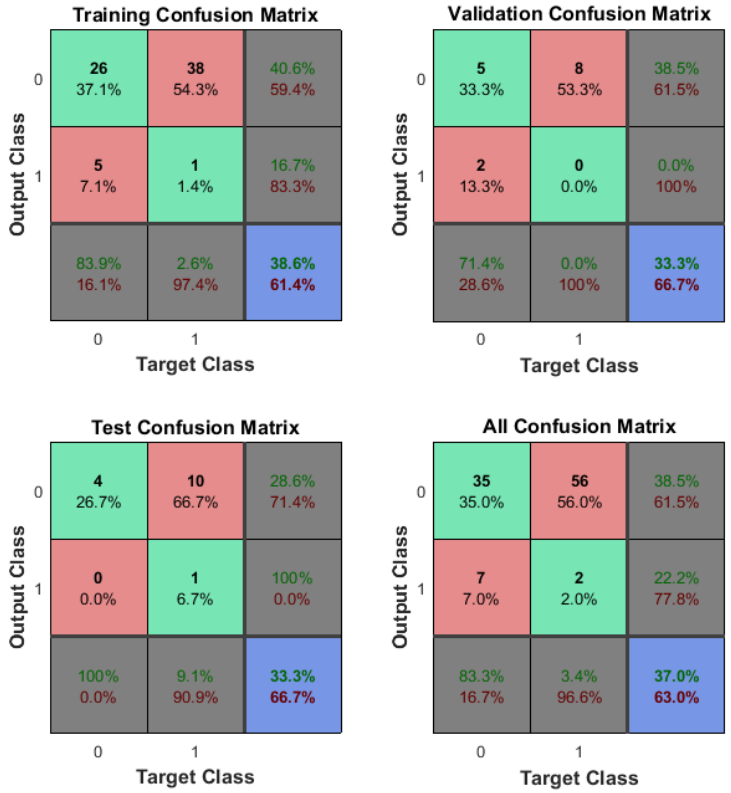
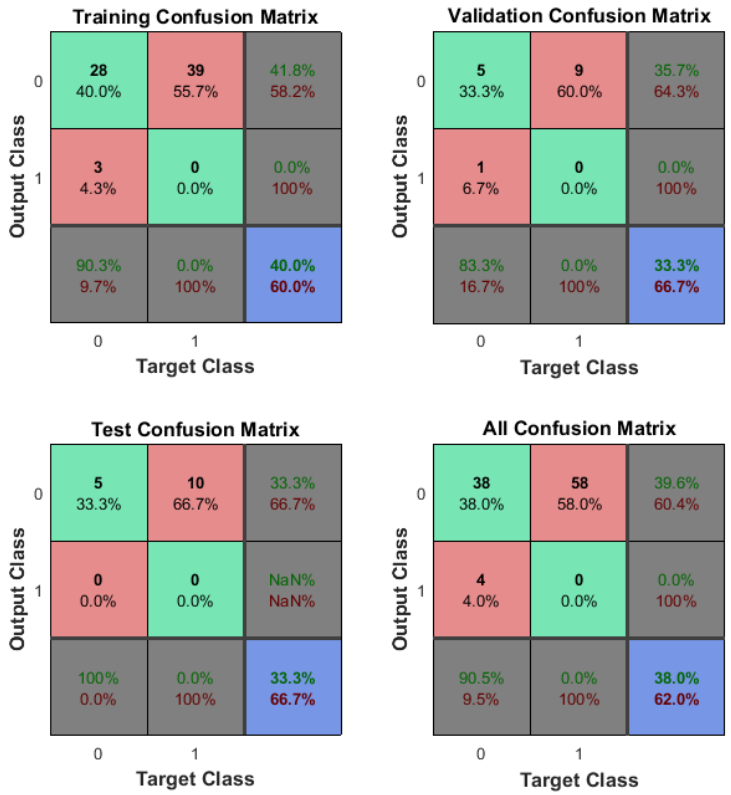
Conj. 2		
10 NEURÔNIOS	SOMA	
	MÉDIA	

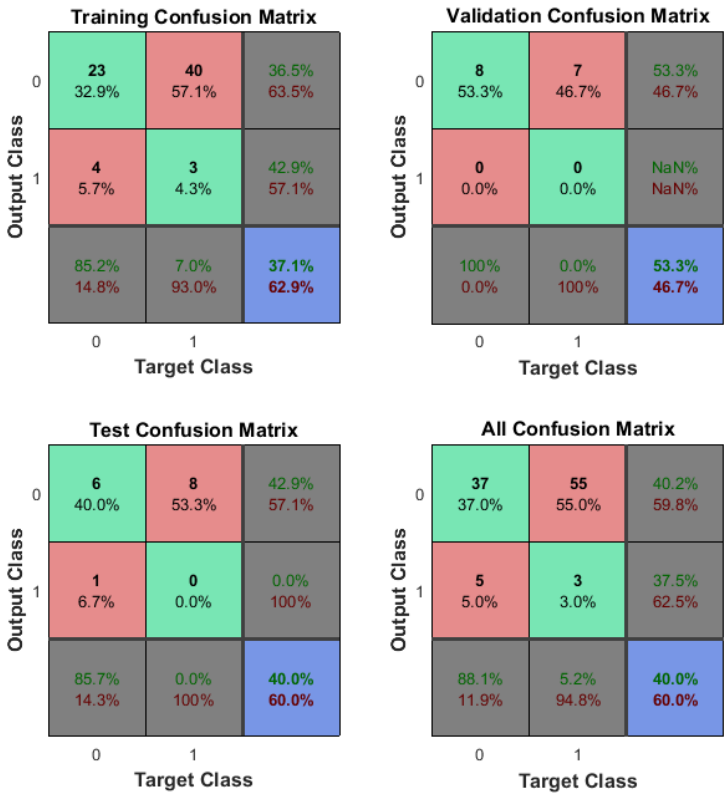
30 NEURÔNIOS	SOMA	
	MÉDIA	

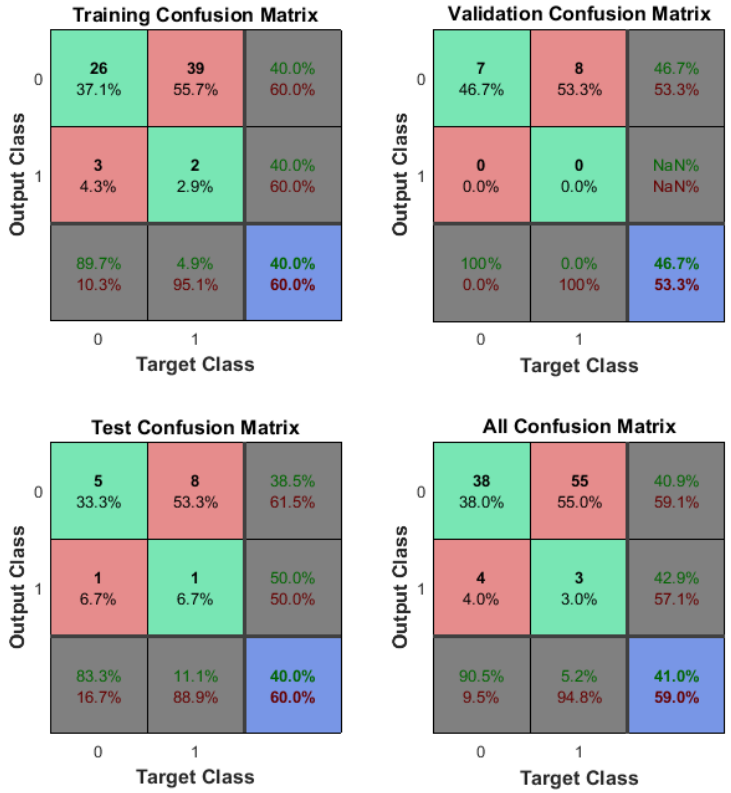
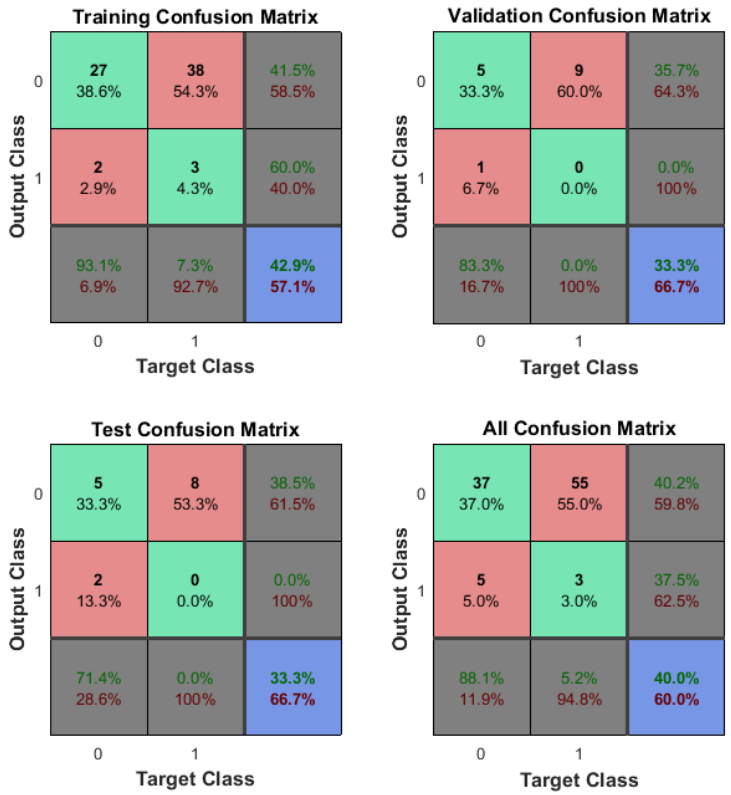
90 NEURÔNIOS	SOMA	
	MÉDIA	

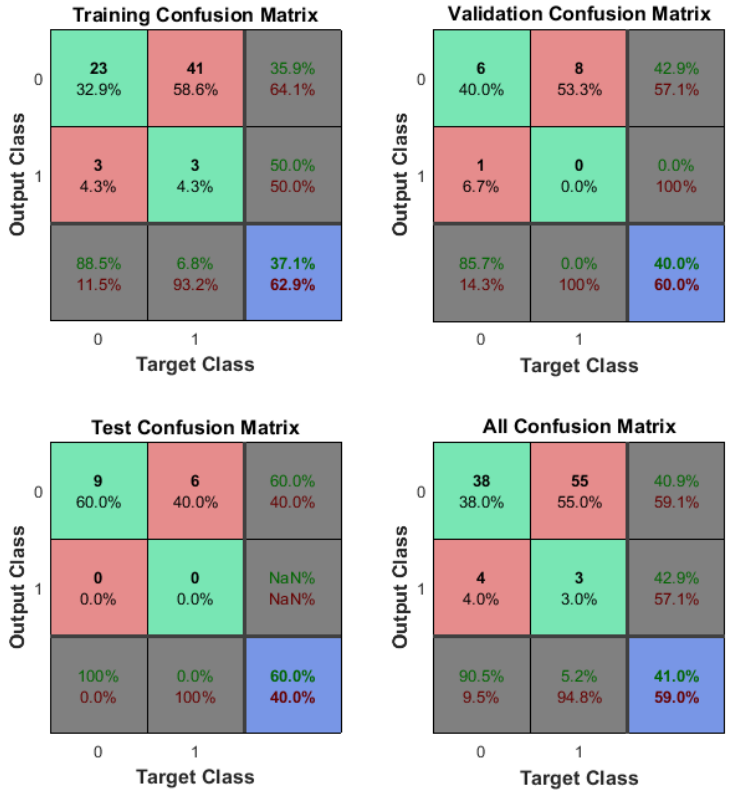
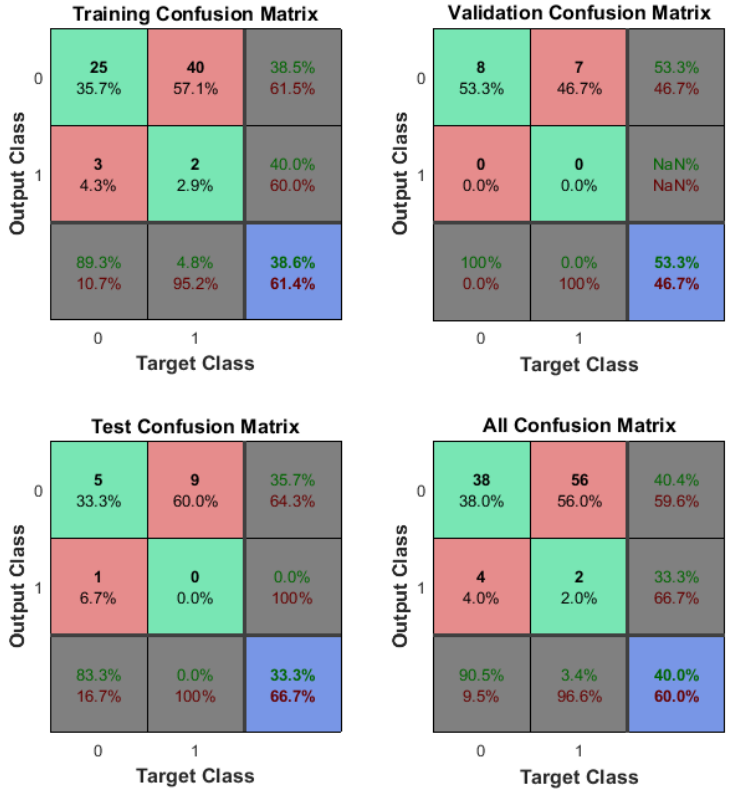
Conj. 3		
10 NEURÔNIOS	SOMA	
	MÉDIA	

30 NEURÔNIOS	SOMA	
	MÉDIA	

90 NEURÔNIOS	SOMA	
	MÉDIA	

Conj. 4		
10 NEURÔNIOS	SOMA	
	MÉDIA	

30 NEURÔNIOS	SOMA	
	MÉDIA	

90 NEURÔNIOS	SOMA	
	MÉDIA	

Nas tabelas acima, como mencionado anteriormente, são mostrados as matrizes de confusão das redes neurais, tanto na utilização da soma dos coeficientes das características, quanto na média. É possível notar que a maioria dos resultados obtidos foram valores aproximados de 50%, sendo esses valores mais concentrados em verdadeiro negativo e falso negativo, mostrando assim que a rede utilizada não foi eficiente, tanto para a soma quanto para a média, tendo em vista que o clássico problema do lançamento de uma moeda nos daria o mesmo resultado.

Este problema pode ter ocorrido por alguns motivos levantados abaixo:

- A *toolbox* utilizada através do aplicativo só permite a modificação de um parâmetro, que é a quantidade de neurônios na camada escondida, fazendo com que a estrutura da rede fique travada, impossibilitando, por exemplo, a utilização de uma função de ativação mais adequada para o problema (padrão utilizado pela *toolbox* é a função sigmoidal).
- A quantidade de dados (neste caso, de imagens) pode ser pequena comparada com a quantidade necessária para um bom resultado da rede (por exemplo, no conjunto 2, onde temos uma quantidade de imagens maior, o resultado foi melhor). Porém, em uma utilização pratica de autenticação de assinaturas, a quantidade de dados utilizada para o reconhecimento de padrões é naturalmente pequena.
- As características extraídas podem não serem as ideais para a rede, sendo necessárias outras características.
- A rede utilizada pode não ser adequada para o problema abordado, sendo necessária a busca por outras implementações (rede RBF, Hopfield, etc).

Por fim, pode-se dizer que utilizar a soma ou média dos coeficientes não afetou muito o resultado tendo em vista que para um mesmo conjunto de dados com uma quantidade de neurônios a soma foi melhor, e para outra quantidade a média foi melhor. Uma questão interessante é que, para todos os conjuntos de dados, às redes neurais utilizadas forneceram uma porcentagem muito baixa de falsos positivos. Falso positivo é quando uma

assinatura falsa é autenticada com verdadeira comprometendo assim o sistema.

8.2. SVM

Abaixo, são mostradas as tabelas referentes aos resultados com a utilização dos diferentes tipos de *kernel* na SVM e variação de seus parâmetros, relacionados à quantidade de erros obtidos na validação do sistema, e falsos positivos (FP) [6]:

Conj 1				
	Soma dos Coeficientes das Características			
Kernel	Valor do Parâmetro	Qtd de Erros	Falsos Positivos	Porcentagem de Acerto
Linear	----------	10	7	83,3
Quadrático	----------	18	9	70,0
Polinomial	3	10	6	83,3
	4	16	9	73,3
	5	19	10	68,3
RBF	1	12	9	80
	2	9	3	85
	3	9	2	85
	4	10	1	83,3
	5	12	3	80
MLP	[0.5-0.5]	28	18	53,3
	[1-1]	25	17	58,3
	[2-2]	27	15	55,0
	[3-3]	26	14	56,7
	[4-4]	26	14	56,7
	Média dos Coeficientes das Características			

[6] Quantidade de valores inferidos pela SVM como verdadeiro (1), quando o resultado correto seria falso (0)

Linear	----------	3	3	95,0
Quadrático	----------	15	11	75
Polinomial	3	10	9	83,3
	4	24	18	60,0
	5	16	13	73,3
RBF	1	11	3	81,7
	2	7	5	88,3
	3	8	7	86,7
	4	7	6	88,3
	5	7	6	88,3
MLP	[0.5-0.5]	24	16	60
	[1-1]	24	18	60
	[2-2]	21	12	65
	[3-3]	20	13	66,7
	[4-4]	23	0	61,7

Conj 2				
	Soma dos Coeficientes das Características			
Kernel	Valor do Parâmetro	Qtd de Erros	Falsos Positivos	Porcentagem de Acerto
Linear	----------	21	17	72,0
Quadrático	----------	23	22	69,3
Polinomial	3	23	19	69,3
	4	31	21	72,0
	5	22	17	70,7
RBF	1	19	2	74,7
	2	21	10	72,0
	3	20	16	73,3
	4	18	16	76,0
	5	18	16	76,0
MLP	[0.5-0.5]	33	25	56,0
	[1-1]	33	24	56,0

	[2-2]	33	25	56,0
	[3-3]	33	25	56,0
	[4-4]	30	23	60,0
	Média dos Coeficientes das Características			
Linear	----------	23	17	69,3
Quadrático	----------	25	22	66,7
Polinomial	3	16	12	78,7
	4	31	19	72,0
	5	17	14	77,3
RBF	1	23	3	69,3
	2	24	15	68,0
	3	20	17	73,3
	4	20	19	73,3
	5	21	20	72,0
MLP	[0.5-0.5]	29	24	61,3
	[1-1]	31	26	72,0
	[2-2]	26	13	65,3
	[3-3]	39	33	48,0
	[4-4]	39	33	48,0

Conj 3				
	Soma dos Coeficientes das Características			
Kernel	Valor do Parâmetro	Qtd de Erros	Falsos Positivos	Porcentagem de Acerto
Linear	----------	4	2	86,7
Quadrático	----------	9	5	70,0
Polinomial	3	4	3	86,7
	4	10	6	66,7
	5	4	3	86,7
RBF	1	10	0	66,7
	2	6	0	80,0
	3	6	1	80,0

	4	6	1	80,0
	5	6	1	80,0
MLP	[0.5-0.5]	11	1	63,3
	[1-1]	12	5	60,0
	[2-2]	10	1	66,7
	[3-3]	12	1	60,0
	[4-4]	21	0	30,0
	Média dos Coeficientes das Características			
Linear	----------	7	3	76,7
Quadrático	----------	9	5	70,0
Polinomial	3	5	4	83,3
	4	5	5	83,3
	5	7		76,7
RBF	1	13	2	56,7
	2	9	2	70,0
	3	8	2	73,3
	4	8	3	73,3
	5	9	3	70,0
MLP	[0.5-0.5]	15	2	50,0
	[1-1]	18	4	40,0
	[2-2]	19	3	36,7
	[3-3]	18	3	40,0
	[4-4]	18	2	40,0

Conj 4				
	Soma dos Coeficientes das Características			
Kernel	Valor do Parâmetro	Qtd de Erros	Falsos Positivos	Porcentagem de Acerto
Linear	----------	18	5	40,0
Quadrático	----------	22	4	26,7
Polinomial	3	20	3	33,3
	4	20	3	33,3

	5	18	1	40,0
RBF	1	23	6	23,3
	2	21	6	30,0
	3	19	6	36,7
	4	19	6	36,7
	5	19	6	36,7
MLP	[0.5-0.5]	16	4	46,7
	[1-1]	18	5	40,0
	[2-2]	18	5	40,0
	[3-3]	22	5	26,7
	[4-4]	22	5	26,7
	Média dos Coeficientes das Características			
Linear	----------	18	4	40,0
Quadrático	----------	18	5	40,0
Polinomial	3	15	3	50,0
	4	21	5	30,0
	5	18	3	40,0
RBF	1	20	4	33,3
	2	21	7	30,0
	3	22	7	26,7
	4	21	6	30,0
	5	21	6	30,0
MLP	[0.5-0.5]	16	4	46,7
	[1-1]	17	4	43,3
	[2-2]	16	4	46,7
	[3-3]	17	4	43,3
	[4-4]	15	5	50,0

Nas tabelas acima, são apresentados todos os resultados obtidos para os quatro conjuntos de dados para os *kernels* Linear, Quadrático, Polinomial (ordens 3, 4 e 5), RBF (sigma 1, 2, 3, 4 e 5) e MLP (params [0.5-0.5], [1-1], [2-2], [3-3] e [4,4]).

Abaixo é mostrada uma tabela com os melhores resultados para cada conjunto de dados:

		Kernel	Valor do Parâmetro	Erros	Porcentagem de Acerto
Conj 1	Soma	RBF	Sigma: 2	9 Erros (2FP)	85,0
	Média	Linear	----------	3 Erros (3 FP)	95,0
Conj 2	Soma	RBF	Sigma: 4	18 Erros (16 FP)	76,0
	Média	Polinomial	Ordem: 3	12 Erros (12 FP)	78,7
Conj 3	Soma	Linear	----------	4 Erros (2 FP)	86,7
	Média	Polinomial	Ordem: 3	5 Erros (4FP)	83,3
Conj 4	Soma	MLP	[0.5-0.5]	16 Erros (4 FP)	46,7
	Média	Polinomial	Ordem: 3	15 Erros (3 FP)	50,0

Nota-se que, utilizando a soma, os dois conjuntos que possuíam uma maior quantidade de imagens (1 e 2) foram os que necessitaram de um *kernel* mais elaborado para um melhor resultado (RBF). Isso pelo fato de que em um conjunto maior de imagens pode haver uma maior variação de características nestas imagens, que podem espalhar os vetores suporte pelo espaço, aumentando a dimensionalidade do espaço dos vetores suporte.

Já utilizando a média, para os conjuntos 2, 3 e 4 o *kernel* polinomial de ordem 3 foi que obteve uma menor quantidade de erros mostrando que ao utilizar a média das características da imagem pode ser uma melhor representação, tendo em vista que um *kernel* polinomial de ordem 3 não é um *kernel* muito elaborado.

Em geral, a utilização da média mostrou-se mais eficiente do que a soma.

9. DISCUSSÃO FINAL

Técnicas e tecnologias de reconhecimento de padrões tornaram-se indispensáveis nos dias atuais, considerando a era da informação.

As técnicas que foram utilizadas neste projeto são técnicas já consagradas para reconhecimentos de padrões. Com isso, o estudo comparativo, que é o estudo proposto por esse projeto, se mostra muito necessário e viável, tendo em vista que nos dias atuais buscam-se as técnicas que façam um reconhecimento de forma precisa (com o mínimo de erros) e de forma rápida (sem necessidade de muito poder computacional). Porém, pelo fato de uma das técnicas utilizadas (Rede Neural) não apresentar resultados satisfatório, a técnica restante (SVM) se mostrou muito eficiente para o propósito proposto.

Para testes dos sistemas implementados, a utilização de bases de dados já disponíveis se mostraram necessárias, pelo fato de que esses bancos possuem uma variedade de assinaturas com características distintas, uma vez que os sistemas deveriam funcionar para toda e qualquer assinatura, além de reduzir o tempo que seria necessário para coleta da quantidade de assinaturas necessárias para treinamento e teste dos sistemas.

Dois algoritmos foram testados: a Rede Neural MLP e a SVM. Como visto, a Rede Neural utilizada não se mostrou eficiente para o problema abordado, sendo as possíveis razões a estrutura da rede da *toolbox* através do aplicativo ser fechada, ou seja, não é possível alterar parâmetros; quantidade de dados (imagens) pequena para uma boa adequação da rede; rede pode ser não adequada para o problema.

Já a SVM se mostrou bastante eficiente para todos os conjuntos de dados, e foi possível mostrar que utilizar a média dos valores das características extraídas foi mais eficiente do que a soma. Os dados utilizados levam em conta

que as assinaturas forjadas pela própria pessoa devem ser consideradas como genuína, para evitar possíveis fraudes. O sistema foi testado também sem a utilização das assinaturas forjadas pela própria pessoa. Com isso, notou-se que não houve alteração na taxa de acerto, ou seja, essas assinaturas não confundiram o sistema, por isso foram apresentados somente os resultados utilizando todas as assinaturas.

Por fim, é possível afirmar que o sistema desenvolvido se mostrou eficiente tendo em vista a pouca quantidade de dados utilizados para treinamento, e poderia ser utilizado na prática em sistemas reais, realizando alguns ajustes para obtenção de uma taxa de erro adequada para o propósito desejado.

10. REFERÊNCIAS

[1] https://blog.gemalto.com/security/2011/09/05/three-factor-authentication-something-you-know-something-you-have-something-you-are/ - Acessado em 07/07/2017

[2] JAIN, Urmila A.; PATIL, Nitin N. A comparative study of various methods for offline signature verification. In: **Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014 International Conference on**. IEEE, 2014. p. 760-764.

[3] DEORE, Madhuri R.; HANDORE, Shubhangi M. Offline signature recognition: Artificial neural network approach. In: **Communications and Signal Processing (ICCSP), 2015 International Conference on**. IEEE, 2015. p. 1708-1712.

[4] KUMAR, Pradeep; SINGH, Shekhar; GARG, Ashwani; PRABHAT, Nishant. Hand written signature recognition & verification using neural network. **International Journal of Advanced Research in Computer Science and Software Engineering**, v. 3, n. 3, 2013.

[5] SERDOUK, Yasmine; NEMMOUR, Hassiba; CHIBANI, Youcef. An improved artificial immune recognition system for off-line handwritten signature

verification. In: **Document Analysis and Recognition (ICDAR), 2015 13th International Conference on**. IEEE, 2015. p. 196-200.

[6] NEMMOUR, Hassiba; CHIBANI, Youcef. Off-line signature verification using artificial immune recognition system. In: **Electronics, Computer and Computation (ICECCO), 2013 International Conference on**. IEEE, 2013. p. 164-167.

[7] ÖZGÜNDÜZ, Emre; ŞENTÜRK, Tülin; KARSLIGIL, M. Elif. Off-line signature verification and recognition by support vector machine. In: **Signal Processing Conference, 2005 13th European**. IEEE, 2005. p. 1-4.

[8] RIBEIRO, Bernardete; LOPES, Noel; GONCALVES, Joao. Signature identification via efficient feature selection and GPU-based SVM classifier. In: **Neural Networks (IJCNN), 2014 International Joint Conference on**. IEEE, 2014. p. 1138-1145.

[9] HAYKIN, S. Redes Neurais: princípios e prática. 2ª edição. Trad: Paulo Martins Engel. 2000.

[10] CANDÈS, Emmanuel J.; DONOHO, David L. Ridgelets: A key to higher-dimensional intermittency?. **Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences**, v. 357, n. 1760, p. 2495-2509, 1999.

[11] Marcus Liwicki, Muhammad Imran Malik , Linda Alewijnse, Elisa van den Heuvel, Bryan Found,. "ICFHR2012 Competition on Automatic Forensic Signature Verification (4NsigComp 2012) ", Proc. 13th Int. Conference on Frontiers in Handwriting Recognition, 2012.

[12] Marcus Liwicki, Elisa van den Heuvel, Bryan Found, Muhammad Imran Malik. "Forensic Signature Verification Competition 4NSigComp2010 – Detection of Simulated and Disguised Signatures", Proc. 12th Int. Conference on Frontiers in Handwriting Recognition, 2010

[13] https://www.sas.com/pt_br/insights/analytics/machine-learning.html# - Acessado em 05/07/2017

[14] AZEVEDO, Frederico A.C.; CARVALHO, Ludmila R.B.; GRINBERG, Lea T.; FARFEL, José Marcelo; FERRETTI, Renata E.L.; LEITE, Renata E.P; JACOB-FILHO, Wilson; LENT, Roberto; HERCULANO-HOUZEL, Suzana. Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. **Journal of Comparative Neurology**, v. 513, n. 5, p. 532-541, 2009.

[15] BEAR, Mark F.; CONNORS, Barry W.; PARADISO, Michael A. **Neurociências: desvendando o sistema nervoso**. Artmed Editora, 2008.

[16] COUTINHO, Eluã Ramos; SILVA, Robson Mariano; DELGADO, Angel Ramon Sanchez. Utilização de Técnicas de Inteligência Computacional na Predição de Dados Meteorológicos. **Revista Brasileira de Meteorologia**, v. 31, n. 1, p. 24-36, 2016.

[17] CORTES, Corinna; VAPNIK, Vladimir. Support-vector networks. **Machine learning**, v. 20, n. 3, p. 273-297, 1995.

[18] MEYER, David; WIEN, FH Technikum. Support vector machines. **R News**, v. 1, n. 3, p. 23-26, 2001.

[19]https://www.mathworks.com/matlabcentral/answers/313449-background-subtraction-from-the-thermal-images - Acessado em 19/04/2018

[20] DENKER, John S.; GRAF, Hans P.; HENDERSON, Donnie; HOWARD, Richard E.; Hubbard, Wayne E.; JACKET, Lawrence D.; O'GORMAN, Lawrence. **Image skeletonization method**. U.S. Patent n. 5,224,179, 29 jun. 1993.

[21]https://stackoverflow.com/questions/24008024/how-to-identify-boundaries-of-a-binary-image-to-crop-in-matlab - Acessado em 19/04/2018

[22]https://www.mathworks.com/matlabcentral/fileexchange/37950-feature-extraction-using-multisignal-wavelet-transform-decomposition - Acessado em 26/04/2018

[23] https://www.mathworks.com/help/stats/svmtrain.html - Acessado em 26/04/2018

11. APÊNDICE

```
=================================================================
% Multiscale Wavelet Transform feature extraction code by Dr. Rami Khushaba
% Research Fellow - Faculty of Engineering and IT
% University of Technology, Sydney.
% Email: Rami.Khushaba@uts.edu.au
% URL: www.rami-khushaba.com (Matlab Code Section)

% last modified 29/08/2012
% last modified 09/02/2013

function feat = getmswtfeat(x,winsize,wininc,SF)

if nargin < 4
   if nargin < 3
      if nargin < 2
         winsize = size(x,1);
      end
      wininc = winsize;
   end
   error('Please provide the sampling frequency of this signal')
end

datawin = ones(winsize,1);
datasize = size(x,1);
Nsignals = size(x,2);

numwin = floor((datasize - winsize)/wininc)+1;
% allocate memory
feat = zeros(winsize,numwin);
st = 1;
en = winsize;

for i = 1:numwin
   curwin = x(st:en,:).*repmat(datawin,1,Nsignals);
   feat(1:winsize,i) = detrend(curwin);

   st = st + wininc;
   en = en + wininc;
end
dec = mdwtdec('col',feat,J,'db4');

if isequal(dec.dirDec,'c')
   dim = 1;
end
[cfs,longs] = wdec2cl(dec,'all');
level = length(longs)-2;

if dim==1
   cfs = cfs';
   longs = longs';
end
numOfSIGs  = size(cfs,1);
num_CFS_TOT = size(cfs,2);
absCFS   = abs(cfs);
absCFS0   = (cfs);
cfs_POW2 = absCFS.^2;
Energy  = sum(cfs_POW2,2);
```

```
percentENER = 0*ones(size(cfs_POW2));
notZER = (Energy>0);
percentENER(notZER,:) =
100*cfs_POW2(notZER,:)./Energy(notZER,ones(1,num_CFS_TOT));

%% Pre-define and allocate memory
tab_ENER = zeros(numOfSIGs,level+1);
tab_VAR = zeros(numOfSIGs,level+1);
% tab_STD = zeros(numOfSIGs,level+1);
tab_WL = zeros(numOfSIGs,level+1);
tab_entropy = zeros(numOfSIGs,level+1);


%% Feature extraction section
st = 1;
for k=1:level+1
   nbCFS = longs(k);
   en  = st+nbCFS-1;
   tab_ENER(:,k) = mean(percentENER(:,st:en),2);
   tab_VAR(:,k) = var(percentENER(:,st:en),0,2);
   tab_WL(:,k)  =  sum(abs(diff(percentENER(:,st:en)').^2))';          percentENER(:,st:en)  =
percentENER(:,st:en)./repmat(sum(percentENER(:,st:en),2),1,size(percentENER(:,st:en),2))
;
   tab_entropy(:,k)                                    =                                    -
sum(percentENER(:,st:en).*log(percentENER(:,st:en)),2)./size(percentENER(:,st:en),2);
   st = en + 1;
end
feat =([log1p([tab_ENER tab_VAR  tab_WL]) tab_entropy]);
```