CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

Engenharia de Computação

Anna Claudia Almeida Anício

RECONHECIMENTO DE SÍMBOLOS SIGNWRITING

Timóteo

2014

Anna Claudia Almeida Anício

RECONHECIMENTO DE SÍMBOLOS SIGNWRITING

Monografia apresentada ao Curso deEngenharia de ComputaçãodoCentro Federal de Educação Tecnológica de Minas Gerais,campus Timóteo, como requisito parcial para obtenção dotítulo de Engenheira de Computação.

Orientador: Aléssio Miranda Júnior
Coorientador: Douglas Nunes de Oliveira

Timóteo

2014

Anna Claudia Almeida Anício

RECONHECIMENTO DE SÍMBOLOS SIGNWRITING

Monografia apresentada ao Curso deEngenharia de Computação do Centro Federal de Educação Tecnológica de Minas Gerais, campus Timóteo,como requisito parcial para obtenção dotítulo de Engenheira de Computação.

_____

Aléssio Miranda Júnior

_____

Douglas Nunes de Oliveira

_____

Élder de Oliveira Rodrigues

Timóteo, 20 deoutubrode 2014

À Deus, por ser o meu amparo.

RESUMO

Este trabalho realiza um estudo sobre técnicas de Reconhecimento de Padrões a fim de aplicá-las no reconhecimento de símbolos SignWriting.O SignWriting é um sistema de escrita desenvolvido por Valerie Sutton, em 1974, nos Estados Unidos da América e consiste na utilização de símbolos visuais, como forma de representar as configurações de mão, os movimentos, as expressões faciais e os movimentos do corpo das línguas de sinais.Reconhecer Padrões consiste em analisar um determinado conjunto de dados e organizá-los de acordo com padrões. No processo de reconhecimento estabelecido, o conjunto de dados foi composto de imagens SignWriting coletadas através de um aplicativo de desenho desenvolvido para Android, onde o usuário desenha o símbolo correspondente ao sinal SignWriting desejado. Foram utilizadas a biblioteca OpenCV para auxiliar os trabalhos de processamento das imagens coletadas e a extração de características das mesmas, e a ferramenta de software livre Weka para auxiliar no processo de reconhecimento.Ao final, foram utilizados três algoritmos de mineração de dados, sendo uma árvore de decisão, um classificador bayesiano e uma rede neural de múltiplas camadas,onde a rede neural e a árvore de decisão obtiveram melhores resultados em bases de dados distintas. Espera-se, com esse trabalho, queo processo de reconhecimento dos símbolos SignWritingcontribua no estímulo pesquisas, tais como um reconhecedor automático de imagens.

Palavras-chave: Linguagem de Sinais. SignWriting. Reconhecimento de Padrões.

ABSTRACT

This paper conducts a study on pattern recognition techniques in order to apply them in SignWriting symbols' recognition. The SignWriting is a writing system developed by Valerie Sutton in 1974 in the United States of America and is the use of visual symbols as a way of representing the configurations of hand movements, facial expressions and body movements of languages signals. Recognizing patterns is to analyze a given set of data and arrange them according to standards. Established in the recognition process, the data set was composed of SignWriting images collected through a drawing application developed for Android, where the user draws the symbol corresponding to the desired signal SignWriting. The OpenCV library to assist the work of processing the collected images and the extraction of characteristics of them, and the Weka open source tool to aid in the recognition process were used. In the end, three data mining algorithms were used, a decision tree, Bayesian classifier and a multilayer neural network, where the neural network and the decision tree obtained better results in separate databases. Hopefully, with this work, that the process of recognition of symbols SignWriting contribute in stimulating research, such as an automatic recognizer images.

Keywords: Sign Language. SignWriting. Pattern Recognition.

LISTA DE FIGURAS

LISTA DE EQUAÇÕES

LISTA DE CÓDIGOS

LISTA DE QUADROS

LISTA DE GRÁFICOS

LISTA DE SIGLAS

API – Application Programming Interface – Interface de Programação de Aplicativos

ARFF - Atributo-Relação Arquivo

ASL - American Sign Language – Linguagem Americana de Sinais

BSD - Berkeley Software Distribution – Distribuição de Software de Berkeley

DAC – Deaf Action Committee – Diretoria de Ação Cultural

GNU – GNU isNot Unix – GNU Não é Unix

GUI – Graphical User Interface – Interface Gráfica do Usuário

IHC - Interação Humano-Computador

JPEG - Joint Photographic Experts Group - Grupo Conjunto de Especialistas em Fotografia

LIBRAS – Linguagem Brasileira de Sinais

MIT - Massachusetts Institute of Technology – Instituto de Tecnologia de Massachusetts

MLP – MultiLayer Perceptron – Perceptronde Multicamadas

OpenCV - Open Source Computer Vision – Fonte Aberto Visão Computacional

PUC – Pontifícia Universidade Católica

RNA – Redes Neurais Artificiais

RP – Reconhecimento de Padrões

RS – Rio Grande do Sul

SVM - Support Vector Machine – Máquina de Vetores de Suporte

WEKA – Waikato Environment for Knowledge Analysis –Waikato Ambiente para Análise de Conhecimento

SUMÁRIO

1 INTRODUÇÃO

A LIBRAS (Língua Brasileira de Sinais) é a língua oficial de sinais do Brasil, reconhecida pelo então ex-presidente Fernando Henrique Cardoso, publicada pela Lei nº 10.436, de 24 de abril de 2002 e a Lei nº 10.098, de 19 de dezembro de 2002 (AZEREDO, 2006). Em dezembro de 2005, foi regulamentada pelo ex-presidente Luiz Inácio Lula da Silva, por meio do decreto 5626/2005 (Decreto5.689, 22 de dezembro de 2005), sendo reconhecida como meio legal de comunicação e expressão entre as comunidades de pessoas surdas no Brasil. Com isso, a LIBRAS não pode ser mais chamada de uma simples linguagem, ela é uma Língua Oficial usada pelos surdos.

Mesmo sendo considerada uma Língua Oficial, a Libras não possui representação textual oficial. Isso dificulta a comunicação textual das pessoas surdas, uma vez que, na grande maioria dos casos em que queiram estabelecer essa comunicação, elas possuem como alternativa a língua utilizada pelas pessoas falantes, ou seja, para o surdo, uma segunda língua.

O sistema SignWriting, desenvolvido por Valérie Sutton do Deaf Action Commite, da Califórnia, USA (SUTTON, 2002), é uma notação gráfica para a Língua de Sinais. O SignWriting pode registrar qualquer língua de sinais do mundo sem passar pela tradução da língua falada. Cada língua de sinais vai adaptá-lo à sua própria ortografia (STUMPF, 2005). Trata-se, portanto, de uma alternativa, caso a pessoa surda deseje aprender uma representação escrita para língua de sinais.

Reconhecimento de Padrões (RP) é entendido como a caracterização de dados de entrada em classes identificáveis através de extração de características ou atributos fundamentais. Característica (ou Atributo) é o dado extraído de uma amostra por meio de medida e/ou processamento (LOPES, 2012).

Reconhecimento de Padrões trata-se, portanto, de uma área de pesquisa que tem por objetivo a classificação de objetos (padrões) em um número de categorias ou classes (THEODORIDIS; KOUTROUMBAS, 1999).

Este trabalho consiste em um estudo sobre técnicas computacionais de Reconhecimento de Padrões, a fim de serem aplicadas no reconhecimento dos símbolos SignWriting escritos à mão. Para isso, serão utilizados como auxílio a ferramenta de software livre Weka (WEKA) e a biblioteca OpenCV (OPENCV) para o processamento e extração das características das imagens SignWriting coletadas através de um aplicativo de desenho desenvolvido para Android (MIRANDA, 2014).

1.1 Motivação

A Língua de Sinais é uma língua gestual eainda não possui representação escrita/textual oficial. Sendo assim,caso uma pessoa surda necessite recorrer à escrita, na grande maioria dos casos, elarecorre à representação escrita da língua sonora utilizada pelas pessoas falantes ou aos seus recursos.

Por exemplo, no Brasil, um surdo, na grande maioria dos casos, necessita aprender o português escrito, que é a sua segunda língua, para escrever algo, pois não existe representação textual para a LIBRAS que é a sua linguagem natural. Nesta segunda língua, o surdo encontra muita dificuldade de expressão, o que faz com que sua produção escrita seja quase inexistente, limitando-se a comunicações básicas, ainda assim efetuadas com dificuldade (STUMPF, 2000).

Portanto, o SignWriting é uma proposta para que as pessoas surdas escrevam em sualíngua materna, ou com uma conotação o mais próximo possível dela, sem terem de usar uma segunda língua, como a utilizada pelas pessoas falantes, por exemplo.A principal motivação deste trabalhoé contribuir com pesquisas na área de reconhecimento de padrões, como as de reconhecedores automáticos de imagens e, através delas, auxiliar no uso da linguagem de sinais também na forma escrita pelos usuários surdos.

1.2 Objetivo

O objetivo desse trabalho éestabelecerum processo para o reconhecimento das imagens SignWriting utilizando técnicas distintas de Reconhecimento de Padrões, a fim de auxiliar na criação de ferramentas úteis e inovadoras para que a comunidade surda faço uso.

Para isso, será utilizado um aplicativo de desenho desenvolvido para Android cuja função será coletar as imagens SignWriting que irão compor a base de dados. Uma vez coletadas, serão extraídas suas características e realizado o processamento das imagens, com o auxílio da biblioteca OpenCV. As características extraídas serão então utilizadas para o reconhecimento das imagens com o auxílio da ferramenta Weka.

É importante ressaltar que o SignWriting é um modelo de representação textual da Linguagem de Sinais, porém não adotado oficialmente em muitos países, inclusive no Brasil, embora existem boas perspectivas à sua aderência, conforme será visto em tópicos posteriores. O trabalho visa o reconhecimento dos símbolos do SignWriting, através do uso de técnicas de Reconhecimento de Padrão.

1.2.1 Objetivos Específicos

Segue os objetivos específicos:

- Uso de um aplicativo de desenho desenvolvido para Android, onde o usuário pôde representar os símbolos do SignWriting;
- Coleta de uma base de dados, com os símbolos “desenhados”, para utilização nasdiferentes técnicas de Reconhecimento de Padrões;
- Processamento das imagens coletadas e extração de características das mesmas, tais como histogramas verticais e horizontais, com o auxílio da biblioteca OpenCV;
- Estudo e definição das melhorestécnicas a serem utilizadas para o reconhecimento dos símbolos do SignWriting;
- Estudo da ferramenta de auxílio Weka, seu formato de entrada, os algoritmos que a compõe, a fim de verificar o que melhor se encaixa a natureza do problema;
- Formalização de um processo para o reconhecimento dos símbolos SingWriting, de modo a facilitar o uso dos mesmos como uma proposta para escrever em linguagem de sinais.

2 ESTADO DA ARTE

Este tópico descreve uma pequena introdução histórica sobre o SignWriting e sobre a técnica de Reconhecimento de Padrões, bem como o estado atual destes, suas evoluções e o que tem sido desenvolvido nessas áreas.

2.1 SignWriting

O sistema de representação de línguas de sinais denominado por SignWriting foi criado por Valerie Sutton no ano de 1974.

Os próximos oito parágrafos são um retrato pertinente ao trabalho baseado na descrição de Ronice Müller de Quadros (QUADROS, 2013) sobre a história do SignWriting.

Inicialmente, Valerie, que também era dançarina, criou um sistema para escrever danças, o DanceWriting, e despertou a curiosidade dos pesquisadores da língua de sinais dinamarquesa que estavam procurando uma forma de escrever os sinais, e, na Dinamarca, foi realizado o primeiro registro da criação de um sistema de escrita de línguas de sinais.

Conforme os registros feitos por Valerie Sutton na homepage do SignWriting[1], em 1974, a Universidade de Copenhagen (na Dinamarca) solicitou à Sutton que registrasse os sinais gravados em vídeo cassete - até então, a única forma de registro das línguas de sinais, registro esse que continua sendo uma forma valiosa para a comunidade surda até os dias atuais.

As primeiras formas foram inspiradas no sistema escrito de danças, porém realizando-se as devidas alterações, fato que caracterizou a década de 70 como um período de transição do DanceWriting para o SignWriting.

Em 1977, o Dr. Judy Shepard-Kegl (PhD em Linguística pelo MIT) organizou o primeiro workshop sobre SignWriting para a Sociedade de Linguística de New

[1] signwriting.org

England nos Estados Unidos, no MIT. Nesse mesmo ano, um primeiro grupo de surdos adultos, um grupo do Teatro Nacional de Surdos em Connecticut, nos Estados Unidos, aprenderam o sistema, e também foi publicada a primeira história escrita em SignWriting. Em 1978, foram editadas as primeiras lições em vídeo, e em 1979, Valerie Sutton trabalhou com uma equipe do Instituto Técnico Nacional para Surdos em Rochester, em Nova Iorque, prestando assistência na elaboração de uma série de livretos chamados The Techinical Signs Manual que usaram ilustrações em SignWriting.

Na década de 80, Valerie Sutton apresentou um trabalho no Simpósio Nacional em Pesquisa e Ensino da Língua de Sinais intitulado como uma forma de analisar a Língua de Sinais Americana e qualquer outra língua de sinais sem passar pela tradução da língua falada, e também foi escrito o primeiro jornal à mão.

Depois disso, o SignWriting começou a se desenvolver mais e mais, e de um sistema escrito à mão livre passou a ser um sistema possível de ser escrito no computador.

A evolução do SignWriting apresenta, de certa maneira, características da evolução da escrita, sendo que a atual discussão é quanto a sua padronização, pois se percebeu que a produção escrita dos sinais difere de pessoa para pessoa, cada um escreve como acha que deve ser escrito. Valerie pode comprovar que isso estava acontecendo no primeiro curso de SignWriting ministrado na PUC do RS em Porto Alegre,no ano de 1997, ondecada aluno produzia o mesmo sinal de forma diferente, sendo alguns mais simples ou mais detalhistas do que outros.

A ASL (American Sign Language – Linguagem Americana de Sinais) tem uma longa caminhada em SignWriting e já dispõe de um dicionário bastante rico produzido pelo DAC (Deaf Action Committee - Diretoria de Ação Cultural). Muitas pessoas estão usando SignWriting nos EUA e a tendência natural é de haver uma padronização. Claro que cada língua de sinais vai naturalmente desenvolver uma forma comum de escrever os sinais, conforme as características de cada povo e país.

Ao fim, logo pelo relato, temos que o SignWriting está se desenvolvendo muito rápido e tem se espalhado em vários países do mundo, como pode ser visto na Figura 1. O DAC está oferecendo suporte para o desenvolvimento de Projetos de Alfabetização em SignWriting. Tais projetos envolvem escolas americanas, canadenses e brasileiras.

Figura 1: Países que Utilizam Atualmente o Sistema SignWriting

Fonte:(SIGNWRITING.ORG)

Quanto ao SignWriting no Brasil, foi descoberto,no ano de 1996, na PUC do RS em Porto Alegre, através do Dr. Antônio Carlos da Rocha Costa que formou um grupo de trabalho envolvendo as professoras Marianne Stumpf e Marcia Borba.

Marianne Stumpf é surda e professora na área de computação na Escola Especial Concórdia em Porto Alegre, e trabalha com o SignWriting em algumas turmas, enquanto Marcia Borba se envolve com a parte de pesquisa relacionada à computação.

No Brasil, existem boas perspectivas de dar continuidade à aderência ao SignWriting, uma vez que algumas escolas começam a se interessar e buscar conhecer tal sistema. A Escola Especial Concórdia de Porto Alegre e a Escola Hellen Keller de Caxias do Sul/RS já começaram a aprender como escrever a

LIBRAS eesse é um passo que tende a ser trilhado por muitas outras escolas. O Instituto Nacional de Educação de Surdos no Rio de Janeiro e algumas escolas em São Paulo também já começaram a se interessar pelo SignWriting, e a Federação Nacional de Educação e Integração de Surdos já demonstrou curiosidade.

Em 2007, os estudantes Cristiano Silva Fraga e Gilvania Clemente Viana, da Universidade Católica do Salvador, desenvolveram um software educativo, denominado ABC SignWriting, utilizando a técnica de Redes Neurais, cujo propósito é auxiliar a alfabetização dos surdos. O software foi desenvolvido para a conclusão do curso de Graduação em Informática.

O ABC SignWriting consiste em um software educativo para auxiliar a alfabetização dos surdos, incentivando a escrita na língua de sinais e facilitando o aprendizado do alfabeto da Língua Portuguesa, através do reconhecimento de símbolos do alfabeto SignWriting desenhados pelo usuário (FRAGA; VIANA, 2007).

2.2 Reconhecimento de Padrões

Padrões são unidades de informação que se repetem, ou então, são sequências de informações que dispõem de uma estrutura que se repetem (NAVEGA, 2002).

Reconhecer padrões é uma das atividades mais recorrentes e mais importantes do ser humano. Lendo um texto escrito, por exemplo, os olhos capturam os sinais representados pelos grafemas (letras) da língua, transformam-nos em sinais nervosos, que são reconhecidos (interpretados) pelo cérebro humano. Este último reconhece aqueles signos, palavras, frases e outros elementos, em um crescendo de informações, até chegar ao sentido do texto, ou seja, até entender o texto (RAUBER,1997).

Os homens utilizam constantemente suas habilidades em reconhecer padrões, essencialmente por meio dos sentidos. Assim pode reconhecer rostos, músicas, vinhos, materiais, etc.

Segundo Rauber (RAUBER, 1997), “reconhecimento de padrões é a transformação da informação do nível subsimbólico (sinais), ao nível simbólico (o do conhecimento)”.

Mais genericamente, pode-se dizer que o Reconhecimento de Padrões (RP) é um ramo da Ciência que trata da classificação e da descrição de objetos (MARQUES de SÁ, 2006).

Um sistema completo para Reconhecimento de Padrões consiste basicamente em (GRACIANO, 2007):

- Aquisição e representação dos dados de observação a serem classificados e/oudescritos;
- Um mecanismo de extração e seleção de características para obter de informações a respeito dos dados;
- Um esquema de classificação, descrição ou aprendizagem dos dados a partirdas características extraídas.

Existem trabalhos que envolvem Reconhecimento de Padrões e SignWriting. A maioria deles utiliza a técnica de Redes Neurais, como o estudo feito por Fabiana Rocha em 2003 (ROCHA, 2003). Em seu trabalho, Fabiana utiliza uma Rede Neural BackPropagation para reconhecer os símbolos manuscritos do sistema SignWriting.

Encontra-se também outros estudos no campo de Reconhecimentos de Padrões que utilizam como apoio as ferramentas Weka e OpenCV, como no trabalho desenvolvido por Geovanny de Oliveira Filho em 2013 (FILHO, 2013), que utiliza o OpenCV como biblioteca para manipulação das imagens, e os algoritmos J48 e MultiLayer Perceptron da ferramenta Weka para o processo de classificação do número de pessoas em motos no trânsito; e no trabalho desenvolvido por Amanda Guimarães em 2012 (GUIMARÃES, 2012) que utiliza o OpenCV juntamente com o Weka para auxílio no diagnóstico de câncer de colo uterino.

3 FUNDAMENTOS TEÓRICOS

Neste tópico será abordado os conceitos do SignWriting, de Reconhecimento de Padrões, da biblioteca OpenCV e da ferramenta Weka, seu formato de entrada e os algoritmos que a compõe e que foram utilizados no processo de reconhecimento.Esses são os conceitos intimamente relacionados ao trabalho proposto.

3.1 SignWriting

O SignWriting é um sistema de escrita das línguas gestuais que expressa os movimentos, as forma das mãos, as marcas não-manuais e os pontos de articulação (SILVA, 2013).

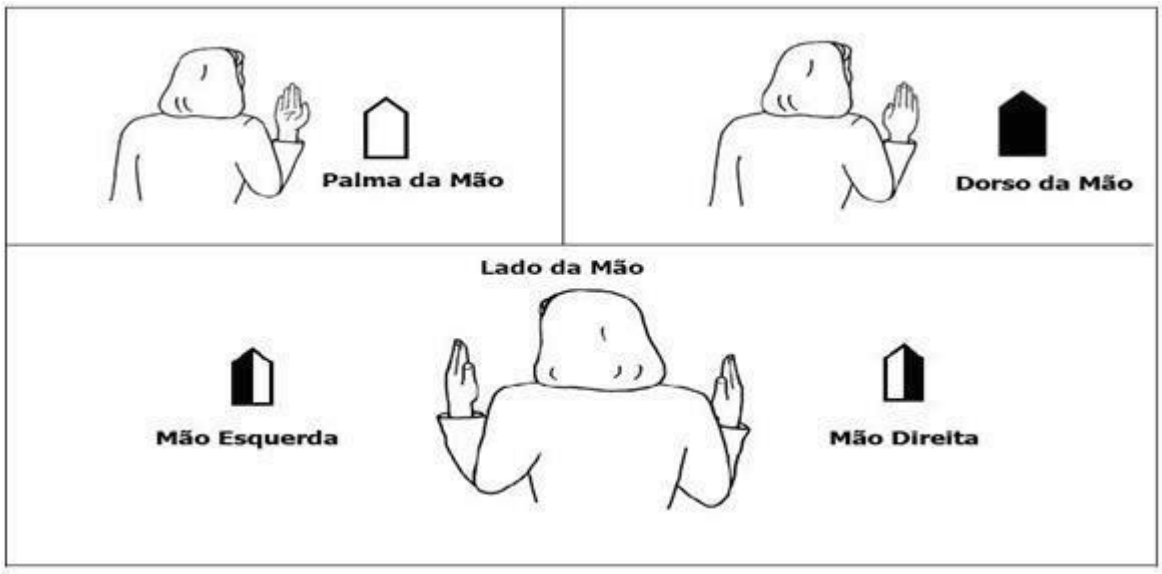
3.1.1 Aplicação Prática

Um discurso em língua gestual é realizado sob o aspecto de duas perspectivas: ado receptor e do emissor. Assim, a primeira diz respeito ao que é observado por alguém que não estáa produzir, mas apenas a visualizar as mãos de alguém que se encontra a gestualizar; por sua vez, asegunda representa aquilo que o próprio gestuante produz e vê. Neste sistema a escrita dos gestos pode ser feita segundo as duas perspectivas, no entanto as publicações em SignWriting são feitas segundo a perspectiva do emissor, sendo a do receptor utilizada ocasionalmente na transcrição de gesto a partir de um vídeo ou quando há necessidade de se fazer um registro rápido que acompanhe o discurso (SUTTON, 2002).

Conforme a perspectiva do emissor quando este vê a palma da mão enquanto produz um gesto o símbolo será branco. Quando o emissor visualiza a parte de trás da própria mão, o dorso, o símbolo será preto. Por sua vez, quando o emissor gestualiza e visualiza o lado da mão, o símbolo será dividido em duas metades, uma

branca e outra preta, isto é, a cor determina a orientação da palma da mão. A Figura 2 ilustra a coloração dos símbolos conforme a perspectiva do emissor.

Figura 2: Perspectiva do Emissor



Fonte: Silva, R. C., 2012

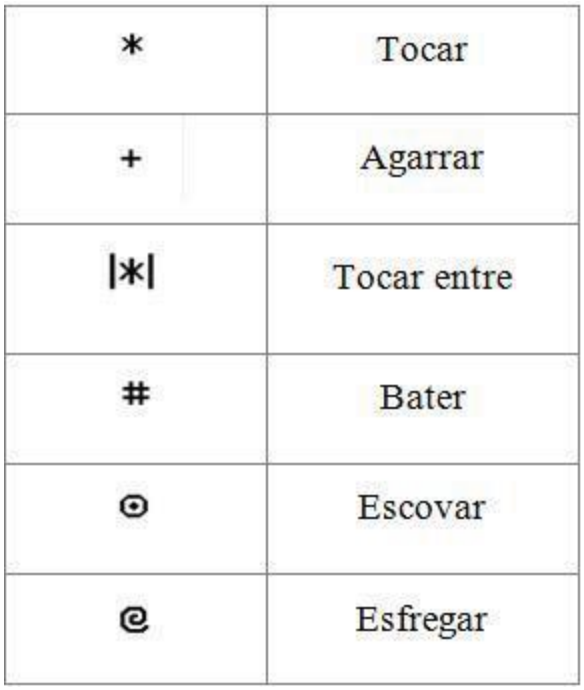
No que diz respeitoàs formas da mão, o SignWriting assenta em três configurações básicas: a mão fechada, amão circular e a mão aberta (Figura3). Todos os outros símbolos que representam as configurações correspondem a variações destes três (STUMPF, 2005).

Figura 3: Configurações Básicas da Mão

Fonte: Silva, R. C., 2012

No SignWriting existem seis formas para representar o contato dos símbolos (Figura 4). Esse contato podeser de uma mão com a outra, da mão com corpo ou da mão com a cabeça.

Figura 4: Símbolos de Contato

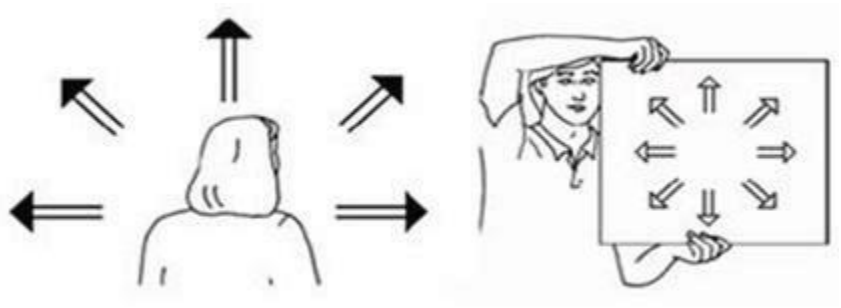


Fonte:Silva, R. C., 2012

O *tocar* é escrito com um asterisco e acontece quando uma mão tem um contato gentil com alguma parte do corpo; o *agarrar* é escrito com uma cruz e é utilizado quando a mão agarra em alguma parte do corpo ou em uma peça de roupa, por exemplo; o *tocar entre* é escrito com um asterisco entre duas linhas verticais e é utilizado quando existe um toque entre duas partes do corpo, habitualmente entre os dedos; o *bater* é escrito com um cardinal e é utilizado quando a mão toca em uma zona com força; o *escovar* é escrito através de um círculo com um ponto preto no meio e é utilizado em contato com movimento que desliza para fora da superfície; por último, o *esfregar*é escrito através de um símbolo espiral e é utilizado num contato com movimento, mas que se mantém na superfície (SUTTON, 2002).

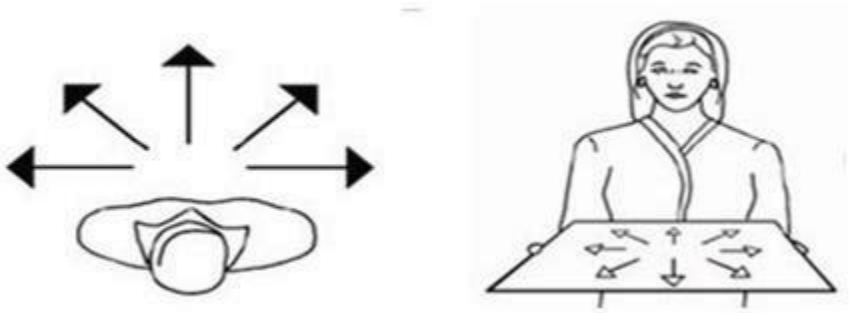
No que diz respeito aos movimentos, eles podem ser classificados em movimentos das mãos ou dos dedos. Para ValerieSutton, ambos se processam num espaço de sinalização que corresponde à área onde se executam os gestos, isto é, a distância que o braço consegue alcançar em frente, cima e baixo. Este espaço divide-se em dois planos que apresentam representações distintas: o plano paralelo à parede, onde os movimentos são para cima e para baixo sendo a representação feita com setas de duplo traço (Figura 5), e o plano paralelo ao chão, em que os movimentos são efetuados para frente e para trás e a representação é feita com setas simples, só com um traço (Figura 6).

Figura 5: Plano Paralelo à Parede



Fonte:Silva, R. C., 2012

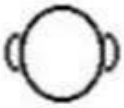
Figura 6: Plano Paralelo ao Chão



Fonte:Silva, R. C., 2012

As componentes não manuais são também expressas no sistema SignWriting. Estas podem ser asexpressões faciais ou a posição do tronco. A expressão facial é baseada no símbolo da face e a partir daí criam-se variações para diferentes representações: sobrancelhas, boca, dentes, bochechas (Figura 7). É importante ressaltar que esse é um dos tópicos de Trabalhos Futuros, portantonão é um ponto de alcancedas extensões desse trabalho, cujo foco são os símbolos básicos, ou símbolos primários.

Figura 7: Expressões Faciais

	Testa
	Sobrancelhas para cima
	Olhos abertos
	Orelhas
	Por o ar para fora
	Boca com sorriso fechado
	Língua lambe o lábio de cima
	Dentes
	Queixo para cima
	Outras partes: pescoço

Fonte: Silva, R. C., 2012

Um dos motivos de escolha de se utilizar os símbolos SignWriting é que esses são símbolos internacionais, o que permite a utilização destes para representargestos de qualquer língua gestual do mundo, sendo que cada língua apenas os adapta à sua própria estrutura (STUMPF, 2005).

A ideia é semelhante aos caracteres indu-arábicos (ou simplesmente arábicos), que correspondem ao sistema de numeração decimal e representam, textualmente, os números falados oralmente. Da mesma forma, o SignWriting é uma representação textual dos gestos produzidos pelos surdos.

3.2 Reconhecimento de Padrões

O Reconhecimento de Padrões é um dos ramos da Inteligência Artificial e trata-se, basicamente, de um sistema capaz de organizar informações de acordo com determinados dados (VALIN, 2009). É entendido como a caracterização de dadosde entrada em classes identificáveis através deextração de características ou atributosfundamentais.

A Inteligência Artificial utiliza-se do Reconhecimento de Padrões para analisar determinado conjunto de dados, também chamado conjunto de treinamento, e organizá-los de acordo com padrões (VALIN, 2009).

Entende-se por padrão as propriedades que possibilitam o agrupamento de objetos semelhantes dentro de uma determinada classe ou categoria, mediante a interpretação de dados de entrada, que permitam a extração das características relevantes desses objetos (TOU; GONZÁLES, 1981). Entende-se por classe de um padrão um conjunto de atributos comuns aos objetos de estudo. Entende-se por características a descrição numérica elementar de um objeto. Assim, reconhecimento de padrões pode ser definido como sendo um procedimento em que se busca a identificação de certas estruturas nos dados de entrada em comparação a estruturas conhecidas e sua posterior classificação dentro de categorias, de modo que o grau de associação seja maior entre estruturas de mesma categoria e menor entre as categorias de estruturas diferentes (CASTRO; PRADO, 1999-2002).

O reconhecimento de padrões visa classificar dados baseados em conhecimento a priori (preliminar ou dedutivo) ou informações estatísticas extraídas de padrões(VALIN, 2009).

Estes padrões a serem classificados normalmente são grupos de medidas ou observações que definem pontos em um espaço multidimensional apropriado. Antes de partir para a análise efetiva, uma etapa de treinamento é realizada: nela o algoritmo de reconhecimento é testado para que seja possível saber se ele encontra os resultados esperados (VALIN, 2009).

Um sistema para Reconhecimento de Padrões engloba três grandes etapas:

- Representação dos dados de entrada e sua mensuração;
- Extração das características;
- Identificação e classificação do objeto em estudo.

Os parágrafos abaixo descrevem essas três etapas segundo Armando de Castro e Pedro Prado (CASTRO; PRADO, 1999-2002).

A primeira etapa refere-se à representação dos dados de entrada que podem ser mensurados a partir do objeto a ser estudado. Essa mensuração deverá descrever padrões característicos do objeto, possibilitando a sua posterior classificação numa determinada classe.

A segunda etapa consiste na extração de características intrínsecas e atributos do objeto. A escolha das características é de fundamental importância para um bom desempenho do classificador. Esta escolha é feita objetivando os fenômenos que se pretende classificar. Exige-se, portanto, um conhecimento específico sobre o problema em estudo.

A terceira etapa em Reconhecimento de Padrões envolve a determinação de procedimentos que possibilitem a identificação e classificação do objeto em uma classe de objetos.

É válido ressaltar também que existem dois tipos de Reconhecimento: o Supervisionado e o Não Supervisionado, porém ambos entram em ação após a

identificação do padrão. O Reconhecimento Supervisionado utiliza o conjunto de treinamento para classificar os dados obtidos de acordo com as categorias já existentes e nelas organizá-los. Por sua vez, o Reconhecimento Não Supervisionado utiliza o conjunto de treinamento para criar novas categorias, ao invés de simplesmente separar os dados de acordo com as categorias já existentes(VALIN, 2009).

Existem inúmeras áreas onde as técnicas de Reconhecimento de Padrões podem ser aplicadas. Dentre elas pode-se destacar:

- Processamento de Sinais de Voz;
- Análise de Cenas;
- Sensoriamento Remoto;
- Reconhecimento e Descrição de Objetos, tais como Escrita e Faces;
- Geologia;
- Biologia;
- Classificação de Documentos e Dados;
- Identificação de Assinaturas.

3.2.1 Tarefas de Aprendizado

Para extrair conhecimento relevante sobre um domínio específico existem diversas técnicas que podem ser utilizadas. Não existe um modelo universal para resolver todas as situações (FAYYAD; PIATETSKY-SHAPIRO; SMYTH,1996). A escolha de uma técnica para resolver uma situação particular de certa forma é uma arte. Portanto é necessário informar qual o problema que se deseja resolver e quais as metas que devem ser alcançadas. As técnicas de aprendizado podem estar caracterizadas como:

- **Associação**: é a descoberta de relações de associação ou correlações entre um conjunto de itens. São frequentemente expressadas na forma de regras que mostram as condições atributo-valor que acontecem frequentemente juntas em um determinado conjunto de dados;
- **Classificação:** analisa um conjunto de dados de treinamento (conjunto de dados cuja classificação já é conhecida) e constrói um modelo para

cada classe baseando nas características dos dados. Uma árvore de decisão ou um conjunto de regras de classificação é gerado por tal processo de classificação, que pode ser usado para entender melhor cada classe no banco de dados e para classificação de futuros dados;

- **Agrupamento:** análise de agrupamentos consiste em identificar possíveis agrupamentos nos dados, onde um agrupamento é uma coleção de objetos que são “semelhantes”;
- **Sequência:** analisa um grande conjunto de dados de séries temporais para encontrar certas regularidades e características interessantes. Descobrindo assim padrões sequenciais, periodicidades, tendências e divergências. Por exemplo, pode-se predizer a tendência dos valores acionários para uma companhia baseando-se sua historia acionaria, situação empresarial, desempenho dos competidores e mercado atual.

Vale ressaltar que nenhuma técnica pode ser considerada o melhor para todas as aplicações. Para a escolha da técnica mais adequada para o problema deve-se conhecer o domínio. Saber quais os dados são mais importantes, conhecer os padrões já existentes e assim por diante.

3.3 Biblioteca OpenCV

O OpenCV (Open Source Computer Vision - Fonte Aberto Visão Computacional) é uma biblioteca de programação, desenvolvido pela Intel, com funções de visão computacional de tempo real. Distribuída sob licensa BSD (Berkeley Software Distribution – Distribuição de Software de Berkeley), é, portanto, gratuito tanto para uso acadêmico quanto comercial. Possui interfaces para C/C++, Phyton e Java, e suporte para Windows, Linux, Android e MacOS.

Andreia Wizbicki e Gerson Battisti citam o OpenCV como uma biblioteca utilizada para desenvolver aplicações capazes de realizar o reconhecimento de padrões em imagens, tendo sido projetada para a eficiência computacional e com um forte foco em aplicações em tempo real (WIZBICKI; BATTIST, 2014).

A biblioteca OpenCV é largamente utilizada em problemas de Reconhecimento de Padrões, tais como reconhecimento facial (MACHADO, 2009),

reconhecimento de mão (BARBOSA; MELO; SOTUYO; MATRAKAS, 2013), reconhecimento de placa veicular (NASCIMENTO, 2012).

A biblioteca é divida em cinco principais tipos de funções (MARENGONI; STRINGHNI, 2009):

1- Processamento de Imagens;
2- Análise Estrutural;
3- Análise de Movimento e Rastreamento de Objetos;
4- Reconhecimento de Padrões;
5- Calibração de Câmera e Reconstrução 3D.

A biblioteca OpenCV oferece suporte para:

- Captura em tempo real;
- Importação de arquivos de vídeo;
- Tratamento básico de imagem (brilho, contraste, limiar);
- Detecção de objetos (rosto, corpo).

As áreas de aplicação do OpenCV incluem:

- Funcionalidades 2D e 3D;
- Reconhecimento facial;
- Reconhecimento de gestos;
- Interação Humano-Computador (IHC);
- Robótica móvel;
- Compreensão de Movimento;
- Identificação de objetos;
- Segmentação e Reconhecimento;
- Visão Estéreo: percepção de profundidade de 2 câmeras;
- Rastreamento de Movimento;
- Realidade Aumentada.

Além disso, para dar suporte a algumas de suas áreas de aplicação, a biblioteca OpenCV também contém algumas implementações de aprendizagem de máquina, dentre as quais pode-se citar:

- Algoritmo do k-vizinho mais próximo;
- Classificador Bayesiano (NaiveBayes);
- Redes Neurais Artificiais;
- Random Forest;
- Support Vector Machine (SVM).

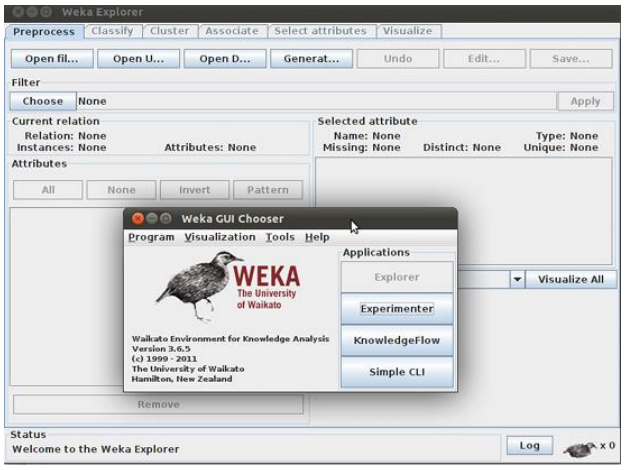
O OpenCV é utilizado hoje em muitos projetos open sources e algumas aplicações comerciais devido a sua boa performance e grande variedade de filtros e algoritmos, sendo a biblioteca open sources mais completa no campo da visão computacional (BRADSKY; PISAREVSKY; BOUGUET, 2006).

3.4 Weka

O Weka (Waikato Environment for KnowledgeAnalysis - Waikato Ambiente para Análise de Conhecimento) é um software de código aberto emitido sob a GNU(GNU is Not Unix – GNU Não é Unix) General Public License.Começou a ser escrito no ano 1993, usando a linguagem de programação Java, na Universidade de Waikato, na Nova Zelândia, e contém uma GUI (Graphical User Interface – Interface Gráfica do Usuário) para interagir com arquivos de dados e produzir resultados visuais. Ele também possui uma API (Application Programming Interface – Interface de Programação de Aplicativos) geral, sendo possível incorporar o Weka, como qualquer outra biblioteca, a aplicativospróprios (WEKA).

A Figura 8 mostra a tela inicial do Weka sobreposta à tela Explorer, ao fundo, que é a opção de maior uso em problemas de Reconhecimento.

Figura 8: GUIWeka



Fonte: (o autor)

O Weka trata-se de uma coleção de algoritmos de aprendizado de máquina para tarefas de mineração de dados. Os algoritmos podem ser aplicados diretamente a um conjunto de dados ou chamados a partir de seu próprio código Java.

O software contém ferramentas para pré-processamento de dados, classificação, regressão, clustering, regras de associação e visualização. Também é bem adequado para o desenvolvimento de novos sistemas de aprendizado de máquina.

O sistema WEKA usa um formato de arquivo comum para armazenar seus conjuntos de dados e, assim, apresenta ao usuário uma visão consistente dos dados. Este formato de arquivo, o formato de Atributo-Relação Arquivo (ARFF), define um conjunto de dados em termos de uma relação ou de mesa feita de atributos ou colunas de dados. As informações sobre os nomes da relação, e os tipos de dados dos atributos são armazenadas no cabeçalho ARFF, os dados propriamente ditos são representados como linhas de dados, no formato de uma matriz, no corpo do arquivo ARFF. A Figura 9 abaixo mostra um exemplo de arquivo ARFF.

Figura 9: Exemplo arquivo ARFF

```
@relation ESDN-weka.filters.unsupervised.instance.NonSparseToSparse

@attribute att1 numeric
@attribute att2 numeric
@attribute att3 numeric
@attribute att4 numeric

@data
{0 3,1 7,3 1}
{2 2,3 8}
```

Fonte: (o autor)

Abaixo serão descritos três dos algoritmos classificadores que compõe a ferramenta Weka e que foram utilizados no processo de Reconhecimento dos símbolos. São eles:

- J48:o algoritmo de Árvore de Decisão;
- NaiveBayes: Classificador Bayesiano Simples;
- MultiLayerPerceptron: Rede Neural de múltiplas camadas.

3.4.1 J48

O J48 é o algoritmo mais popular do Weka, e é uma implementação para o algoritmo C 4.5 para geração top-down de árvore de decisão. Nesta abordagem o atributo mais significativo, ou seja, o mais generalizado, quando comparado a outros atributos do conjunto é considerado raiz da árvore. Na sequência da construção, o próximo nó da árvore será o segundo atributo mais significativo, e, assim, sucessivamente, até gerar o nó folha, que representa o atributo alvo da instância (TAVARES; BONZZA; KONO, 2007).

A árvore de decisão gerada baseia-seno conjunto de dados detreinamento, e tem por finalidade classificar as instâncias no conjunto de teste.

O algoritmo J48 usa uma técnica de busca gulosa, determinando a cadapasso qual o atributo, dentre os disponíveis, que é mais preditivo, e dividindo um nodo da árvore com base nesteatributo.

3.4.2 NaiveBayes

O classificador NaiveBayes é um método probabilístico com base no Teorema de Bayes (MITCHELL, 1997). Este classificador assume que os valores dos atributos

de um exemplo são independentes da sua classe, ou seja, a probabilidade de um evento A (que pode ser uma classe), dado um evento B (que pode ser o valor do conjunto de atributos de entrada), não depende apenas da relação entre A e B, mas também da probabilidade de observar A independentemente de observar B.

A figura 10 abaixo mostra a relação estabelecida pelo Teorema de Bayes,o objetivo é calcular P(A/B), ou seja, calcular a probabilidade de observar A dado que B aconteceu,onde P(A) éa probabilidade a priori de A, P(B) é a probabilidade a priori de B e P(B/A) éa probabilidade de observar B dado que A aconteceu.

Equação 1:Teorema de Bayes

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

Fonte: (o autor)

A probabilidade de ocorrência do evento B pode ser estimada pela observação dafrequência com que esse evento ocorre. Da mesma forma, é possível estimar a probabilidade de ocorrência de um evento B para cada classe A, P(B|A). A dúvida assenta no cálculo da probabilidade de ocorrer A dado B, P(A|B). O Teorema de Bayes resolve este problema utilizando a probabilidade a priori da classe, P(A). Para calcular a probabilidade a priori da classe é necessário manter um contador para cada classe e assumir uma distribuição norma para os atributos.

Os classificadores Bayesianos, do tipo NaiveBayes, calculam a probabilidade de uma instância pertencer a cada uma das classes pré-determinadas, assumindo que há independência entre os atributos que descrevem a instância.A partir destas probabilidades, é possível gerar as saídas com uma regra de decisão que sempre dá como resposta a classe que obteve maior probabilidade após a aplicação do teorema de Bayes (RODRIGUES; AMARAL, 2012).

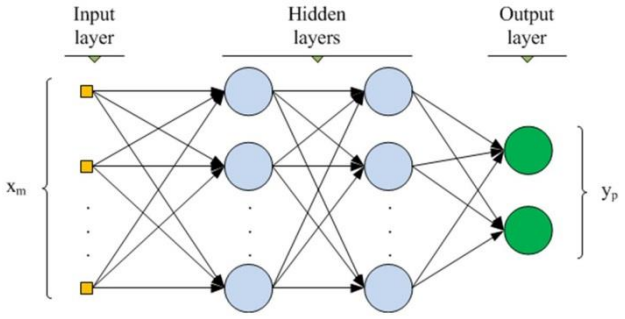
3.4.3 MultiLayerPerceptron (MLP)

As Redes Neurais MLP sãoredes de múltiplas camadas onde cada camada tem uma função específica. A camada de saída recebe os estímulos da camada intermediária e constrói o padrão queserá a resposta. As camadas intermediárias, também chamadas camadas escondidas ou camadas ocultas, são responsáveis pelo aperfeiçoamento da rede, tomando por base o erro retornado na camada de saída.Seus pesos são uma codificação de características apresentadas nos padrõesde entrada e permitem que a rede crie sua própria representação, mais rica e complexado problema. Durante o treinamento com o algoritmo backpropagation, arede opera em uma sequência de dois passos(CARVALHO, 2006):

- Primeiro, um padrão é apresentado à camada de entrada da rede. A atividade resultante flui através da rede, camada por camada, até que a resposta seja produzida pela camada de saída;
- Segundo, a saída obtida é comparada à saída desejada para esse padrão particular. Se esta não estiver correta, o erro é calculado. O erro é propagado a partir da camada de saída até a camada de entrada, e os pesos das conexões das unidades das camadas internas vão sendo modificados conforme o erro é retropropagado.

A figura 10 abaixo ilustra a arquitetura de uma Rede Neural de 2 (duas) Camadas.

Figura 10: Arquitetura de uma MLP com 2 camadas ocultas



Fonte: Geronimo, T. M.; Cruz, C. E. D.; Campos. F. de S.; Aguiar, P. R.; Bianchi E. C., 2014

Numa MLP, o principal parâmetro é o número h de nós da camada oculta (camada intermediária). Por default, o WEKA utiliza uma heurística que corresponde a dividir onúmero de atributos somado ao número de classes por 2 (RAMISCH, 2009).

4 METODOLOGIA

O trabalho realizado é dividido, basicamente, em seis partes. Sendo elas:

1- Coleta das imagens através do uso de um aplicativo de desenho desenvolvido para Android, onde o usuário pode representar os símbolos do SignWriting;

   - Optou-se por um aplicativo para Android devido à precisão dos dedos ser maior que a do mouse, o que torna as imagens mais favoráveis ao uso;
   - Os “desenhos” coletados são destinadosa compor a base de dados que será utilizadapara o treinamento dos diversos algoritmos de Reconhecimento de Padrões.

2- Estudo e uso da biblioteca OpenCV, para o processamento das imagens coletadas, bem como a extração das características;

3- Coleta dos dados de cada imagem obtidos através da geração dos histogramas verticais e horizontais;

4- Geração dos arquivos de extensão ARFF que servirão como entrada da ferramenta Weka;

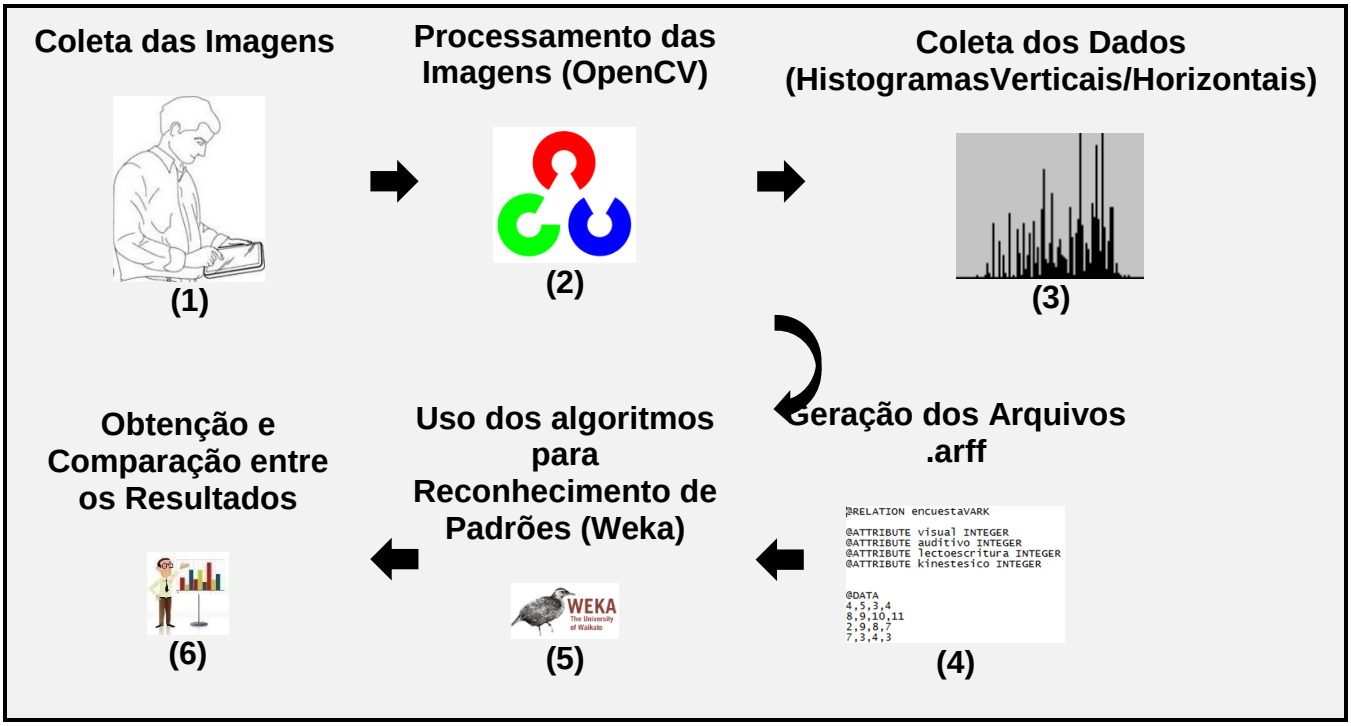
5- Estudo e uso da ferramenta de auxílio Weka, seu formato de entrada, os algoritmos que a compõe, a fim de verificar o que melhor se encaixa a natureza do problema;

   - Neste tópico também foram realizados estudos e testes de algumas técnicas de Reconhecimento de Padrões, de modo a definir quais as melhores a serem utilizadas, qual possuirá um melhor desempenho e atenderá melhor aos objetivos do trabalho, que é o reconhecimento satisfatório das imagens;

6- Obtenção de um processo a fim de obter o reconhecimento dos símbolos SignWriting e a comparação entre os resultados das diferentes técnicas aplicadas.

A figura 11 abaixo ilustraa metodologia adotada no trabalho.

Figura 11: Metodologia Adotada



Fonte: (o autor)

5 RESULTADOS

5.1 Base de Dados

A base de dados foi coletada utilizando-se um aplicativo para desenho desenvolvido para Android, executado em um tablet Samsung GalaxyTab 2 7.0. O aplicativo foi desenvolvido pelo aluno Marcos Guilherme, graduando do curso de Engenharia de Computação do CEFET-MG, campus VII, Timóteo (MIRANDA, 2014).

A base é composta de 80 diferentes símbolos que representam várias combinações de configuração de mão básicas do SignWriting e estes foram replicados em novas imagens, para comporem uma base de dados mais sólida a ser utilizada nas distintas técnicas de Reconhecimento de Padrões. Cada imagem foi replicada 15 vezes, totalizando 1200 imagens que compõem a base de dados.

A base foi dividida em dois grupos, sendo o primeiro composto de imagens coletadas pelo autor e o segundo, imagens coletadas por um grupo de, aproximadamente, 5 pessoas distintas. O primeiro grupo contém a maior parte das imagens (80%), 960 do total das 1200, enquanto o segundo contém as 240 imagens restantes (20%).

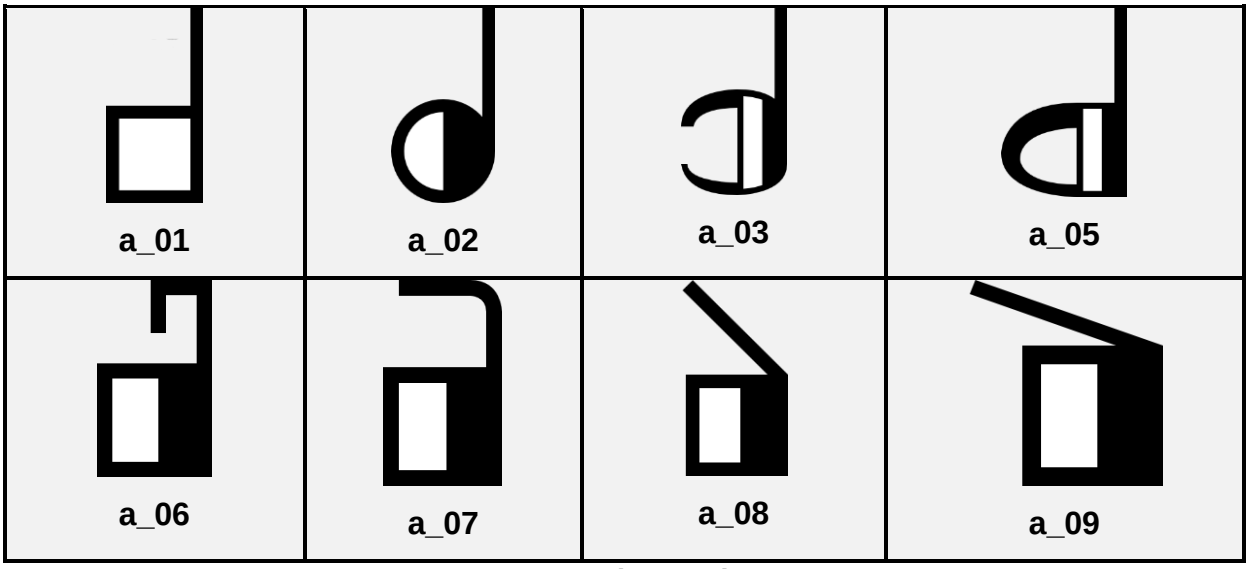
A divisão na base tem o propósito de tornar o processo de reconhecimento mais eficiente, além de estabelecer comparações ao longo desse processo.

A Figura 12 contém alguns dos 80 símbolos que compõe a base de dados. A listagem completa poder ser vista no Anexo I. Percebe-se que as figuras a_06, a_07, a_08 e a _09, possuem a mesma formação básica, diferenciando apenas no movimento do dedo (traço acima da base quadrada). Já as figuras a_01, a_02, a_03 e a_05 possuem formações básicas diferentes. Esses são um dos aspectos que facilitam e/ou dificultam o reconhecimento dos símbolos, uma vez que a chance de erros ao reconhecer figuras semelhantes aumenta.

A Figura 13 contém 3 diferentes coletas do símbolo a_01, coletadas para a Base 01. As imagens da Base 01 foram coletadas pelo mesmo usuário, mesmo assim, percebe-se diferença nos traços, mas um padrão bastante semelhante, como pode ser observado nas figuras (1) e (3).

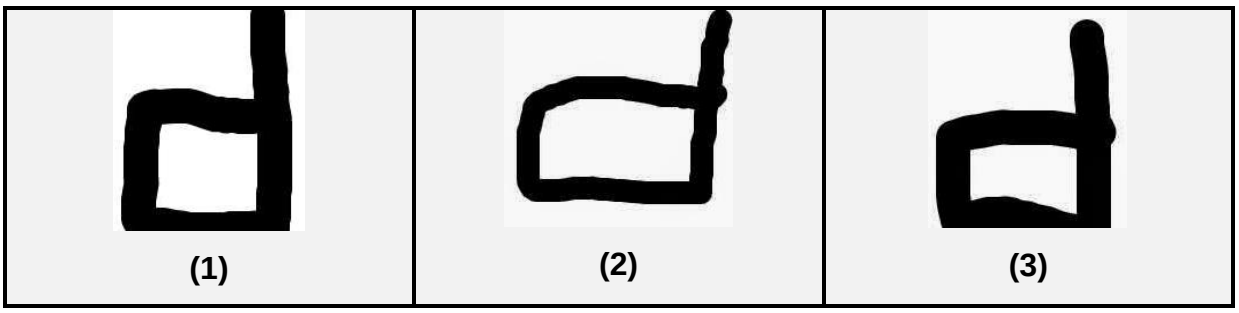
Já a Figura14 contém outras 3 diferentes coletas do mesmo símbolo (a_01), porém coletados para a Base 02. Por se tratar de uma coleta de usuários diferentes, percebe-se diferença nos traços ainda maiores que as observadas na Figura 14.

Figura 12: Exemplos de Símbolos da Base de Dados



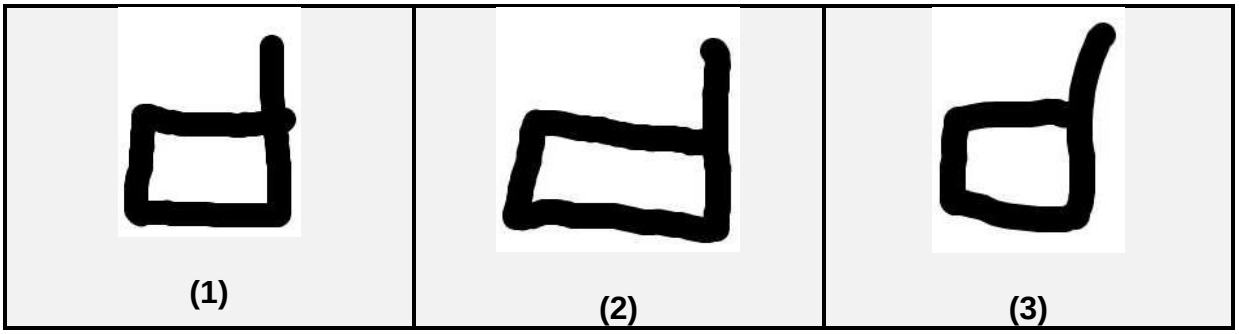
Fonte: (o autor)

Figura 13: Coletas do símbolo a_01 na Base 01



Fonte: (o autor)

Figura 14: Coletas do símbolo a_01 na Base 02



Fonte: (o autor)

5.2 Processamento das Imagens

O processamento das imagens foi realizado com o auxílio da biblioteca OpenCV, descrita no item 3.4, onde foram extraídas as características das imagens que compõe a base de dados, tais como histogramas verticais e horizontais, através de uma aplicação desenvolvida em C++.

A aplicação faz a leitura e interpretação das imagens presentes na base de dados, extrai suas características e gera como saída um arquivo no formato ARFF válido, a fim de ser importado e manipulado pela ferramenta Weka.

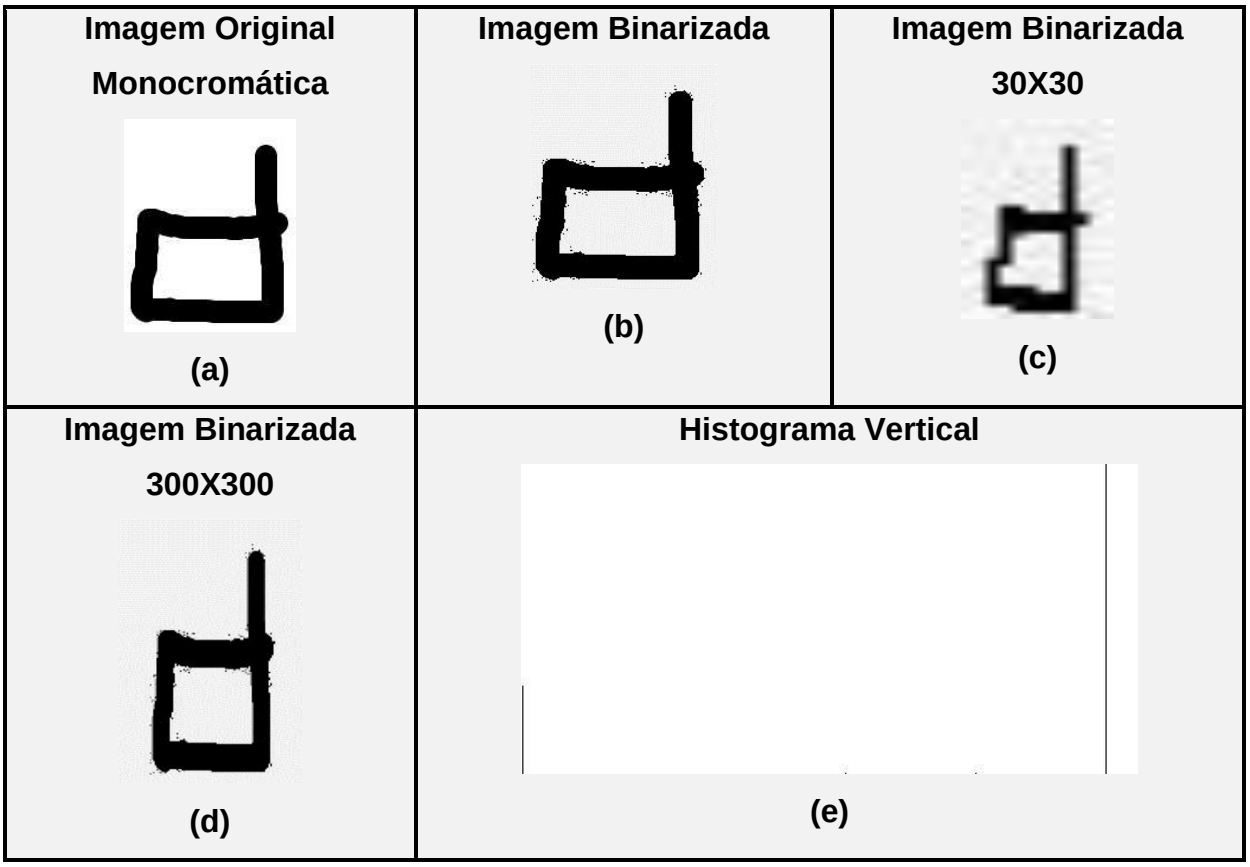
As características extraídas serviram para compor o arquivo ARFF, que é a entrada da ferramenta Weka, descrita no item 3.3, podendo assim aplicar os diversos algoritmos de Reconhecimento que compõe a ferramenta.

A extração de características se deu conforme os seguintes passos:

- É carregada a imagem original, oriunda de um arquivo JPEG (Figura 15 (a));
- A imagem é então binarizada (Figura 15(b));
- A imagem binarizada é então recortada, mantendo apenas a área útil (novas larguras e alturas), sendo então alterada para um tamanho de 30X30 pixels (Figura 15(c));
- A imagem binarizada é novamente redimensionada a um tamanho de 300X300 pixels, gerando uma imagem quadrada grande (Figura 15(d));
- Geração dos histogramas, descrito no item 5.2.1(Figura 15 (e));
- Geração do arquivo ARFF, descrito no item 5.2.2.

As imagens abaixomostram os passos seguidos na extração das características.

Figura 15: Extração das Características



Fonte: (o autor)

5.2.1 Histogramas

O histograma de uma imagem em tons de cinza é uma função que produz o número de ocorrências de cada nível de cinza na imagem, ou seja, para cada nível de intensidade, o histograma representa o número de pixels desse nível, onde o eixo horizontal refere-se à intensidade (0 a 255) e o eixo vertical a quantidade de pixels que apresenta essa intensidade.

Para um melhor aproveitamento das características foram gerados histogramas verticais e horizontais para as duas bases utilizadas.

Os trechos de código 1 a 4 abaixo referem-se à função `histogram()` utilizada para o cálculo dos histogramas. A função recebe como parâmetro um objeto do tipo `IplImage` correspondente à imagem cujo histograma será calculado e retorna outro objeto também do tipo `IplImage` correspondente ao histograma calculado.

Código 1: Função histogram – Declaração das Variáveis

```
1  IplImage* histogram(IplImage *src){
2
3  // Tamanho de um histograma 1D
4  int bins = HIST_WIDTH;
5  inthsize[] = {bins};
6
7  IplImage* imgHistogram = 0;
8  CvHistogram* hist = 0;
9
10 float max_value = 0;
11 float min_value = 0;
12
13 float value;
```

Fonte: (o autor)

Nas linhas 4 e 5, é determinado o tamanho do histograma, cuja variável `HIST_WIDTH` é definida com o valor 256, que corresponde aos 256 tons de cinza, ou seja, o histograma terá 256 intervalos de valores diferentes de níveis de cinza.

Na linha 7 é declarada a variável `imgHistigram` do tipo `IplImage` que será o retorno da função.

Na linha 8 é declarada a variável `hist` do tipo `CvHistogram` que armazenará o histograma criado.

Nas linhas 10 a 13 são declaradas as demais variáveis utilizadas na função.

Código 2: Função histogram – Eixo X do histograma

```
14 // Ranges, escala de cinza: 0-256
15 floatxranges[] = {0, 256};
16 float* ranges[] = {xranges};
```

Fonte: (o autor)

Na linha 15, tem-se a definição da faixa de valores do eixo X do histograma, neste caso, irá variar entre 0 e 256, e na linha 16 um ponteiro é criado para o array que contém a faixa de valores.

Código 3: Função histogram – Cálculo do histograma

```
17  int i;
18
19  IplImage* planes[] = {src};
20
21  hist = cvCreateHist(1, hsize, CV_HIST_ARRAY, ranges, 1);
22  cvCalcHist(planes, hist, 0, NULL);
23  cvGetMinMaxHistValue(hist, &min_value, &max_value);
```

Fonte: (o autor)

Na linha 19 é definida uma imagem do tipo `IplImage` que armazenará os valores correspondentes à imagem passada por parâmetro para a função.

Na linha 21 o histograma é criado com os parâmetros definidos nas linhas anteriores (Código 1 e Código 2) e na linha 22 os valores do histograma são calculados.

Código 4: Função histogram – Desenho do histograma

```
24  if (imgHistogram == NULL)
25  imgHistogram = cvCreateImage(cvSize(bins*2, HIST_HEIGHT), 8, 1);
26
27  // Desenha a area de trabalho do histograma
28  cvRectangle(imgHistogram, cvPoint(0,0),
29  cvPoint(HIST_WIDTH*2,HIST_HEIGHT), CV_RGB(255,255,255), -1);
30
31  // Desenhar as linhas do histograma
32  for (i=1 ; i <= bins ; i++)
33  {
34  // Pega o valor do histograma na posicao i da imagem
35  value = cvQueryHistValue_1D(hist, i-1);
36
37  // Printa a linha do Histograma
38  cvLine(imgHistogram, cvPoint((i*2)-1,HIST_HEIGHT),
39  cvPoint((i*2)-1,HIST_HEIGHT-value), CV_RGB(0, 0, 0));
40
41  makefile<< value << ",";
42  }
43
44  makefile<< "a_" <<endl;
45   return imgHistogram;
46  }
47
```

Fonte: (o autor)

Na linha 24 é criada a imagem onde serão impressos os valores do histograma.

Na linha 29 é desenhada a área de trabalho do histograma.

Nas linhas 33 a 42, as linhas do histograma são desenhadas, onde o valor de cada posição é pego pela função `cvQueryHistValue_1D` (linha 36) e impresso na imagem pela função `cvLine` (linha 39). Na linha 42, o valor pego na linha 36 é impresso em no arquivo de texto `makefile` para a geração do arquivo de dados de extensão `.arff` a ser interpretado pelo Weka.

A linha 45 delimita os valores da nova imagem a serem impressos no mesmo arquivo de dados.

Por fim, na linha 46 ocorre o retorno da função.

Um exemplo de um histograma de uma imagem pode ser observado na figura 14 (e).

5.2.2 Arquivo ARFF

O arquivo de extensão **.arff** é o arquivo responsável pela entrada dos dados na ferramenta Weka. Foram construídos 4 arquivos deste tipo, sendo um arquivo que contem os dados de todos os histogramas verticais e um arquivo que contem os dados de todos os histogramas horizontais, para ambas as bases.

O arquivo ARFF é formado por duas partes, sendo um cabeçalho que contém o nome da relação (nome do arquivo) e os atributos, e os dados propriamente ditos.

Ostrechos de código5 e6 abaixo correspondem ao arquivo gerado com os dados dos histogramas horizontais da Base 02.

Código 5: Arquivo .arff – Cabeçalho

```
1   @RELATION base2hor
2   @ATTRIBUTE pixel-0INTEGER
3   @ATTRIBUTE pixel-1INTEGER
4   @ATTRIBUTE pixel-2INTEGER
5   @ATTRIBUTE pixel-3INTEGER
... ...
256 @ATTRIBUTE pixel-254 INTEGER
257 @ATTRIBUTE pixel-255 INTEGER
258 @ATTRIBUTE classe
    {a_01,a_02,a_03,a_05,a_06,a_07,a_08,a_09,a_10,a_11,a_12,a_13,a_14,a_1
    7,a_18,a_21,a_22,a_23,a_24,a_25,a_26,a_27,a_28,a_29,a_30,a_31,a_32,a_
    33,a_34a,a_34b,a_35,a_36,a_37,a_38,a_40,a_41,a_43,a_44,a_46,a_47,a_48
    ,a_49,a_50,a_51,a_52,a_53,a_54,a_55,a_57,a_58,a_61,a_62,a_63,a_67,a_6
    8,a_71,a_73,a_74,a_75,a_76,a_78,a_79,a_80,a_82,a_83,a_84,a_85,a_86,a_
    87,a_88,a_92,a_93a,a_96,a_99,a_100,a_102,a_103,a_104,a_105,a_106}
```

Fonte: (o autor)

As linhas 1 a 258 correspondem ao cabeçalho do arquivo.

Na linha 1 é definido o nome do arquivo, através do atributo `@RELATION`.

Nas linhas 2 a 257 são definidos os 256 atributos correspondentes aos 256 valores gerados pela função `histogram()` para cada um dos 256 tons de cinza da imagem. Todos eles são do tipo inteiro (classificados como atributos numéricos) e a ordem da declaração indica a posição de cada atributo na seção `@DATA`.

Na linha 258 é definido o atributo classe, que corresponde ao conjunto dos 80 símbolos que representam as configurações de mão, cujas imagens deverão ser classificadas. Este é um atributo classificado como categórico, ou seja, é preciso fornecer uma lista indicando todos os valores do atributo.

Código 6: Arquivo .arff – Dados

```
260  @DATA
261  233,0,0,0, ... ,0,0,0,654,0,0,0, ... ,0,0,0,a_28
262  253,0,0,0, ... ,0,0,0,642,0,0,0, ... ,0,0,0,a_51
263  195,0,0,0, ... ,0,0,0,692,0,0,0, ... ,0,0,0,a_57
264  228,0,0,0, ... ,0,0,0,660,0,0,0, ... ,0,0,0,a_106
...  ...
498  239,0,0,0, ... ,0,0,0,641,0,0,0, ... ,0,0,0,a_46
499  222,0,0,0, ... ,0,0,0,663,0,0,0, ... ,0,0,0,a_36
```

Fonte: (o autor)

A linha 260 corresponde a marcação `@DATA` que indicará à ferramenta onde iniciam-se a leitura dos dados.

As linhas 261 a 499 contem os dados propriamente dito das 240 imagens que compõem a base de dados. Cada linha contém 256 valores correspondentes a cada um dos atributos declarados no cabeçalho do arquivo, onde estes devem aparecer na ordem em que são declarados no cabeçalho, e o atributo de número 257 corresponde a qual classe a imagem deverá ser classificada. Também esta deverá estar de acordo com o conjunto de classes definido nas linhas 16 a 22. São, no total,240 linhas de dados, sendo uma linha para cada imagem da base de dados.

5.3 Reconhecimento dos Símbolos

Foram escolhidos três algoritmos classificadores presentes na ferramenta Weka para o processo de reconhecimento dos símbolos SignWriting, sendo eles um algoritmo para gerar Árvore de Decisão, o J48; um Classificador Bayesiano Simples, o NaiveBayes; e uma Rede Neural de MúltiplasCamadas, MultiLayer Perceptron (MLP), cujas descrições podem ser observadas nos itens 3.4.1, 3.4.2, 3.4.3, respectivamente.

Os algoritmos foram rodados para os dados obtidos através da geração dos histogramas verticais e horizontais das duas bases em que as amostras foram divididas.

É importante ressaltar que a escolha dos algoritmos se deu através da leitura de trabalhos relacionados, da frequência de uso destes, bem como do sucesso de aplicação em problemas semelhantes.

Os itens 5.3.1, 5.3.2 e 5.3.3 abaixo, correspondem a saída do Weka relativa ao índice de acertos/erros do retorno de cada algoritmo para os dados dos histogramas horizontais para a Base 02. Uma descrição detalhada da saída Weka para a mesma base de dados pode ser vista no Apêndice C.

5.3.1 J48

Como descrito no item 3.4.1, o algoritmo J48 é um algoritmo para geração de árvore de decisão. O trecho de código abaixo corresponde a um sumário com os resultados obtidos.

O algoritmo J48 obteve 40,7407% das instâncias da base de teste corretamente e 59,2593% classificadas incorretamente (linhas 174 e 175, respectivamente), como pode ser visto no código 7.

Código 7: Saída J48 – Sumário

```
171  === Evaluation on test split ===
172  === Summary ===
173
174  Correctly Classified Instances          33               40.7407 %
175  Incorrectly Classified Instances        48               59.2593 %
176  Kappa statistic                          0.3955
177  Mean absolute error                      0.015
178  Root mean squared error                  0.1041
179  Relative absolute error                 60.6039 %
180  Root relative squared error             93.5727 %
181  Total Number of Instances               81
```

Fonte: (o autor)

5.3.2 NaiveBayes

Como descrito no item 3.4.2, o algoritmo NaiveBayes é um método probabilístico com base no Teorema de Bayes. O trecho de código abaixo corresponde a um sumário com os resultados obtidos.

O algoritmo NaiveBayes obteve 48,1481% das instâncias da base de teste classificadas corretamente e 51,8519% classificadas incorretamente (linhas 1561 e 1562, respectivamente), como pode ser visto no código 8.

Código 8: Saída NaiveBayes – Sumário

```
1558 === Evaluation on test split ===
1559
1560 === Summary ===
1561 Correctly Classified Instances          39               48.1481 %
1562 Incorrectly Classified Instances        42               51.8519 %
1563 Kappa statistic                          0.4708
1564 Mean absolute error                      0.0128
1565 Root mean squared error                  0.1036
1566 Relative absolute error                 52.0554 %
1567 Root relative squared error             93.1665 %
1568 Total Number of Instances               81
```

Fonte: (o autor)

5.3.3 MultiLayerPerceptron (MLP)

Como descrito no item 3.4.3,as redes MultiLayer Perceptron (MLP) são redes neurais de múltiplas camadas. O trecho de código abaixo corresponde a um sumário com os resultados obtidos.

O algoritmo MLP obteve 53,0864% das instâncias da base de teste classificadas corretamente e 46,9136% classificadas incorretamente (linhas 57451 e 57452, respectivamente), como pode ser visto no código 9.

Código 9: Saída MLP – Sumário

```
57448 === Evaluation on test split ===
57449 === Summary ===
57450
57451 Correctly Classified Instances          43               53.0864 %
57452 Incorrectly Classified Instances        38               46.9136 %
57453 Kappa statistic                          0.5227
57454 Mean absolute error                      0.0139
57455 Root mean squared error                  0.0907
57456 Relative absolute error                 56.141  %
57457 Root relative squared error             81.5375 %
57458 Total Number of Instances               81
```

Fonte: (o autor)

5.3.4 Resultados

O quadro abaixo reúne os dados dos três algoritmos utilizados (J48, NaivesBayes e MLP) para cada uma das amostras (histogramas verticais e horizontais das Bases 01 e 02), com as respectivas taxas de acertos e erros de cada algoritmo.

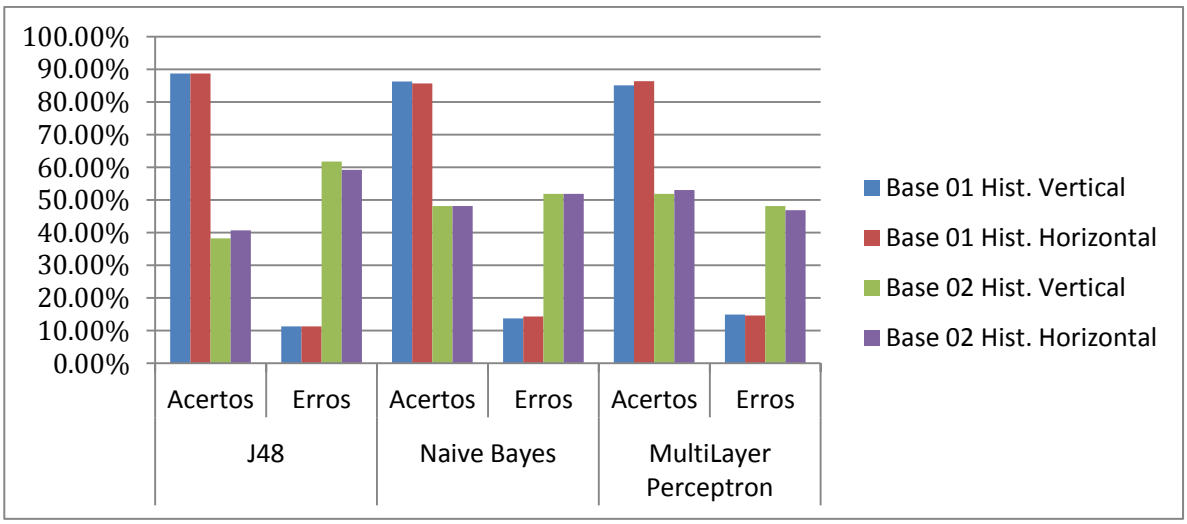
Quadro 1: Acertos/Erros

		J48		NaiveBayes		MultiLayerPerceptron	
		Acertos	**Erros**	**Acertos**	**Erros**	**Acertos**	**Erros**
Base 01	**Hist. Vertical**	88,7195%	11,2805%	86,2805%	13,7195%	85,061%	14,939%
	Hist. Horizontal	88,7195%	11,2805%	85,6707%	14,3293%	86,3659%	14,6341%
Base 02	**Hist. Vertical**	38,2716%	61,7284%	48,1481%	51,8519%	51,8519%	48,1481%
	Hist. Horizontal	40,7407%	59,2593%	48,1481%	51,8519%	53,0864%	46,9136%

Fonte: (o autor)

Os mesmos percentuais podem ser vistos também no Gráfico 1:

Gráfico 1: Acertos/Erros



Fonte: (o autor)

O índice de acertos na Base 01 para todos os métodos foi significativamente maior do que na Base 02. Tal estatística já era esperada, uma vez que a Base 01

possui quatro vezes mais a quantidade de elementos (símbolos SignWriting) do que a Base 02.

O maior percentual de acertos da Base 01 se deu com o algoritmo para geração de árvore de decisão J48, que fora o pior percentual de acertos para a Base 02. Enquanto o maior percentual de acertos da Base 02 se deu com o algoritmo de redes MLP, que também fora o pior percentual de acertos para a Base 01.

A estatística observada para a Base 02 era de fato a esperada, diferentementedos resultados revelados para a Base 01. Acredita-se que o fato de a Base 01 ser composta de uma maior quantidade de dados e esses dados terem sidos manipulados pela mesma pessoa, tenha tornado-a uma base de dados viciada, interferindo assim nos resultados.

6 CONCLUSÃO

A LIBRAS é a língua de sinais oficial do país, utilizada pela comunidade surda, no entanto, por se tratar de um contexto gestual, ainda não existe representação textual/escrita para tal.O sistema SingnWriting aliado às técnicas de Reconhecimento de Padrões facilita esse processo, pois trata-se de uma alternativa para se escrever Linguagem de Sinais.

Para isso, foram escolhidas duas ferramentas de auxílio: a ferramenta gráfica OpenCV, utilizada na extração das características; e o software livre Weka, utilizado para o reconhecimento em si. Ambas as ferramentas ofereceram grande ajuda no desenvolvimento do projeto, por já trazerem consigo aplicações específicas da área de Reconhecimento de Padrões, facilitando o processo metodológico.

Foram escolhidos três algoritmos classificadores:

- J48: para a geração de Árvores de Decisão;
- NaiveBayes: Classificador Bayesiano Simples;
- Rede MLP: rede neural de Múltiplas Camadas.

Os algoritmos foram escolhidos por se tratarem de naturezas distintas (uma árvore de decisão, um classificador lógico e uma rede neural, respectivamente), e por obterem sucesso em literaturas da área.

Os três algoritmos foram aplicados nos dados extraídos dos histogramas verticais e horizontais das duas bases coletadas.

Na base que possuía uma menor quantidade de elementos, conforme o esperado, a Rede MLP obteve uma maior taxa de acerto, 51,8519% para os histogramas verticais e 53,0864% para os histogramas horizontais. Enquanto que na base que possuía uma maior quantidade de elementos, a Árvore J48 obteve uma maior taxa de acerto, 88,7195% tanto para os histogramas verticais quanto para os histogramas horizontais.Acredita-se na chance da base de dados maior ser uma base viciada, uma vez que os dados foram coletados pela mesma pessoa.

Apesar da chance de existir uma base viciada na primeira base, e um pequeno número de símbolos na segunda, obteve-se o processo de Reconhecimento das imagens SignWriting, passando pela coleta dos dados,

extração das características e identificação, conforme visto na teoria de Reconhecimento de Padrões, no item 3.2.

A coleta dos dados abrange o recolhimentodas 1200 imagens correspondente à 80 diferentes símbolos que representam várias combinações de configuração de mão básicas do SignWriting. As imagens foram e divididas em dois grupos a fim de obter análises e comparações entre eles no processo de reconhecimento. A coleta deu-sepelo aplicativo de desenho SignBank desenvolvido para Android.

A extração das característicasabrange à obtenção dos histogramas verticais e horizontais de cada uma das 1200 imagens que compõem as bases de dados. Para isso, foi utilizado a biblioteca open source OpenCV, largamente recomendada por diversas literaturas da área. A biblioteca também foi utilizada no tratamento das imagens, antes da extração das características das mesmas.

Com os dados extraídos dos histogramas das imagens, foram gerados os arquivos de extensão ARFF, responsável pela entrada dos dados na ferramenta Weka. Esse arquivo é o responsável pela identificação das classes, uma vez que nele contem os registros de todos os histogramas da base de dados determinada, bem como a indicação à qual classe pertence.

Por fim, a identificação corresponde à última etapa do processo estabelecido. Nessa etapa foram rodados três algoritmos da ferramenta de software livre Weka. A função de cada algoritmo é identificar a taxa de erros e acertos para cada uma das bases, de acordo com o arquivo ARFF de entrada.

O processo obtido foi satisfatório prometendo evolução na taxa de acertos à medida que os atritos encontrados forem ajustados.

7 TRABALHOS FUTUROS

Como continuação desse trabalho, sugere-se ampliar a base de dados e estabelecer novas divisões e classificações internas, tais como idade de quem as desenhou, divisão por grupos de símbolos primários originários, dentre outras.

Pode-se realizar também testes com outros algoritmos de Reconhecimento incorporados à própria ferramenta Weka, ou a outra ferramenta ou, até mesmo, à implementações próprias. A fim de obter resultados ainda mais satisfatórios e comparações ainda mais precisas.

Pode-se também aumentar a base de dados e extrair outros diferentes tipos de características das imagens. A fim, também, de tornar o Reconhecimento ainda mais preciso e satisfatório.

Outro trabalho que também poderá ser realizado a prazos futuros é o reconhecimento de expressões faciais, tais como sobrancelhas, boca, dentes, bochechas, dentre outras, que também existem no alfabeto do SingWriting, bem como de figuras compostas e de movimentos, uma vez que o foco deste trabalho foi o reconhecimento dos símbolos primários, ou símbolos básicos.

REFERÊNCIAS

AZEREDO, E., **Língua Brasileira de Sinais** “Uma Conquista Histórica”, Brasília, 2006. Disponível em: <http://www.prefeitura.sp.gov.br/cidade/secretarias/upload/pessoa_com_deficiencia/arquivos/Libras_Uma_conquista_historica.pdf>. Acesso em: 23 ago. 2013.

BARBOSA, B., C.; MELO, J., R. de;SOTUYO, J., A.; MATRAKAS, M., D.,**Visão Computacional Aplicada para o Reconhecimento da Mão como Marcador Virtual**,Foz do Iguaçu, 2013. Disponível em: <http://udc.edu.br/v3/udcanglo/producoes/SEICOM2013/files/Artigo01.pdf>. Acesso em: 10 out. 2014.

BRADSKY, G. R.; PISAREVSKY, V.; BOUGUET, J., **Learning OpenCV: Computer Vision with the OpenCV Library**. Springer, Estados Unidos, 2006.

BRASIL, Decreto5.689, de 22 de dezembro de 2005,**DECRETO Nº 5.626, DE 22 DE DEZEMBRO DE 2005**, Brasília, 2005.Disponível em:<http://www.planalto.gov.br/ccivil_03/_ato2004-2006/2005/decreto/d5626.htm>. Acesso em: 06 out. 2014.

CARVALHO, C., E., S., de, **Reconhecimento de Caracteres Manuscritos Utilizando Redes Neurais Artificiais**,Brasília, 2006. Disponível em:<http://repositorio.uniceub.br/bitstream/123456789/3195/2/9972772.pdf>. Acesso em: 10 out. 2014.

CASTRO, A. A. M. de, PRADO, P. P. L. do,**Algoritmos para Reconhecimento de Padrões**. **Rev. Cienc. Exatas**, Taubaté, v. 5-8,p. 129-145, 1999-2002. Disponível em: <http://site.unitau.br//scripts/prppg/exatas/downloads/algoritmosreconhecimento-99-02.pdf>. Acessoem: 05 fev. 2014.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P., **The KDD Process for Extracting Useful Knowledge from Volumes of Data**, 1996.

FILHO, G. O.,**Classificação do Número de Pessoas nas Motos em Imagens de Trânsito Corretamente Segmentadas**, Quixadá, 2013. Disponível em: <http://www.repositoriobib.ufc.br/000012/00001221.pdf>. Acesso em: 10out. 2014.

FRAGA, C. S.; VIANA, G. C., **ABC SignWriting:** Um Software para Auxiliar na Alfabetização dos Surdos Utilizando uma Rede Neural Artificial, Salvador, 2007. Disponível em: <http://info.ucsal.br/banmon/Arquivos/Mono_060707.PDF>. Acesso em: 23 ago. 2013.

GERONIMO, T. M.; CRUZ, C. E. D.; CAMPOS. F. de S.; AGUIAR, P. R.; BIANCHI E. C., **MLP and ANFIS Applied to the Prediction of Hole Diameters in the Drilling Process**, Bauru, SP. Disponível em: <http://www.intechopen.com/books/artificial-neural-networks-architectures-and-applications/mlp-and-anfis-applied-to-the-prediction-of-hole-diameters-in-the-drilling-process>. Acesso em: 06 out. 2014.

GRACIANO, A. B. V., **Rastreamento de Objetos Baseado em Reconhecimento Estrutural de Padrões**, São Paulo, 2007.

GUIMARÃES, A. S., **Utilizando Características Morfológicas de Tecidos Histológicos para Auxílio ao Diagnóstico de Câncer de Colo Uterino**, Ribeirão Preto, 2012. Disponivel em:<http://dcm.ffclrp.usp.br/ibm/upload/amanda.pdf>. Acesso em: 10 out. 2014.

LOPES, F., M., **Introdução ao Reconhecimento de Padrões e Aplicações em Problemas de Bioinformática**, 2012. Disponivel em:<http://www.ime.usp.br/posbioinfo/cv2012/reconhecimentoPadroes_FabricioLopes.pdf>. Acesso em: 10 out. 2014.

MACHADO, B., B., Implementação de um Algoritmo de Reconhecimento Facial UsandoEigenFace, Belo Horizonte, 2009. Disponível em: <http://revistas.unibh.br/index.php/dcet/article/view/247/137>. Acesso em: 10 out. 2014.

MARENGONI, M; STRINGHNI,D, **Introdução a Visão Computacional usando OpenCV**, 2009.

MARQUES de SÁ, J. P., **Reconhecimento de Padrões**, Faculdade de Engenharia da Universidade do Porto, 2000. Disponível em:<http://paginas.fe.up.pt/~jmsa/recpad/>. Acesso em: 11 jul. 2014.

MIRANDA, A., **SignBankMobile – A Data Collection Environment for Deaf Culture Handwriting Recognition System**, 2014

MITCHELL, T., M., **Machine Learning**,1997, Pag. 5 - 9.

NASCIMENTO, J., D. do, **Detecção e Reconhecimento de Placa Automotiva com Baixo Custo**, Brasília, 2012. Disponível em: <http://repositorio.uniceub.br/bitstream/235/3665/2/Monografia%20JEAN%20DIAS%202_2012.pdf>. Acesso em: 10 out. 2014.

NAVEGA, S., **Princípios Essenciais do DataMining**, São Paulo, 2002.Disponível em: <http://www.intelliwise.com/reports/i2002.htm>. Acesso em: 10 out. 2014.

OPENCV, OpenCV (Open Source Computer Vision), Disponívelem: <http://opencv.org/>. Acesso em: 12 jul. 2014.

QUADROS, R. M. de, Um capítulo da história do SignWriting. In **A History of SignWriting Written in Brazilian Portuguese**. Disponível em: <http://www.signwriting.org/library/history/hist010.html>. Acesso em: 09ago. 2013.

RAMISCH, C., **Trabalho Prático de Mineração de Dados - Algoritmos de Aprendizado para Avaliação de Carros**, 2009. Disponível em: <http://www.inf.ufrgs.br/~ceramisch/download_files/courses/Undergraduate_BRAZIL/UFRGS_2009_1/Topicos_Especiais_em_Computacao_I_-_Mineracao_de_Dados_-_INF01179/Trabalho_1_-_Car_Evaluation/Relatorio.pdf>. Acesso em: 10 out. 2014.

RAUBER, T. W.,**Reconhecimento de Padrões**. Mini-Curso.JOURNEY OF ACTUALIZATION IN COMPUTER SCIENCE XVII.CONGRESS OF THE BRAZILIAN COMPUTER SCIENCE SOCIETY, Brasília, DF, 1997.

ROCHA, F., Z., F.,**Proposta de um Padrão Manuscrito para Reconhecimento Automático dos Símbolos do Sistema SignWriting (SW)**, Pelotas, 2003. Disponível em: <http://www.signwriting.org/archive/docs1/sw0081-BR-Monografia.pdf>. Acesso em: 10 out. 2014.

RODRIGUES, F., A.; AMARAL, L., R., do,**Aplicação de Métodos Computacionais de Mineração de Dados na Classificação e Seleção de Oncogenes Medidos por Microarray**, 2012. <http://www.inca.gov.br/rbc/n_58/v02/pdf/14_artigo_aplicacao_metodos_computacionais_mineracao_dados_classificacao_selecao_oncogenes_medidos_microarray.pdf>. Acesso em: 10 out. 2014.

SILVA, R. C. da, **SignWriting: Um Sistema de Escrita das Línguas Gestuais**: Aplicação à Língua Gestual Brasileira, Escola Superior de Educação de Coimbra, dez. 2012. Disponível em: <http://www.exedrajournal.com/exedrajournal/wp-content/uploads/2013/01/31-numero-tematico-2012.pdf>. Acesso em: 25 jul. 2013.

STUMPF, M. R., **Linguaguem de Sinais: escrita dos surdos na Internet**. In V CONGRESSO IBEROAMERICANO DE INFORMÁTICA EDUCATIVA, 2000, ViñaDelmar, Chile. Disponível em: <http://www.ufrgs.br/niee/eventos/RIBIE/2000/papers/031.htm>. Acesso em: 28 jul. 2013.

STUMPF, M. R.,**Aprendizagem de escrita de língua de sinais pelo sistema signwriting:** línguas de sinais nopapel e no computador, Porto Alegre, 2005. Disponível em: <http://www.lume.ufrgs.br/bitstream/handle/10183/5429/000515254.pdf?sequence=1>. Acesso em: 05 ago. 2013.

SUTTON, V., **Lições sobre oSignWriting**: Um Sistema de Escrita para Língua de Sinais. Tradução de Marianne Rossi Stumpf. Colaboração de Antônio Carlos da Rocha Costa e Ronice Muller de Quadros. DAC – DeafActionCommitte for SignWriting, 2002.

TAVARES, C.; BONZZA, D.; KONO, F., Descoberta de Conhecimento Aplicado a Dados Eleitorais, 2007. Disponível em: <http://gc.facet.br/v5n1/pdf/descoberta_de_conhecimento_aplicado_a_dados_eleitorais.pdf> . Acesso em: 10 out. 2014.

THEODORIDIS, S.; KOUTROUMBAS, K., **Pattern Recognition**,1999.

TOU, J. T., GONZÁLEZ, R. C., **Pattern Recognition Principles**, Massachusetts, 1981.

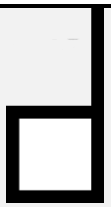
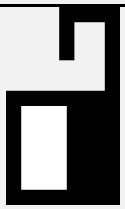
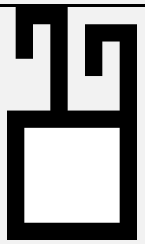
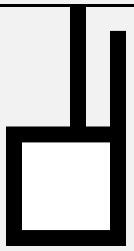
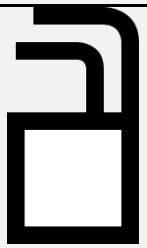
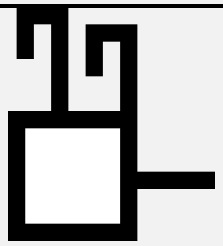
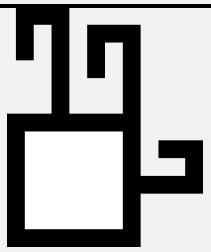
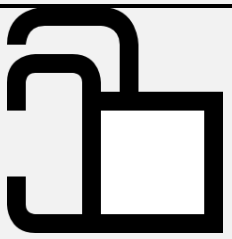
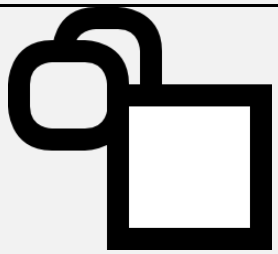
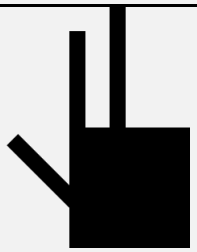
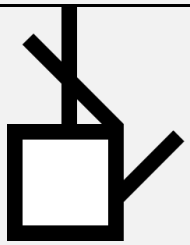
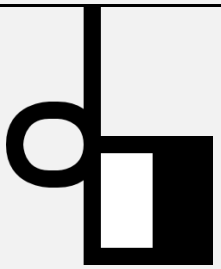
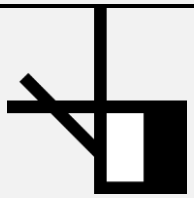
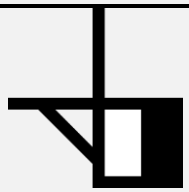
VALIN, A., **Inteligência Artificial:** Reconhecimento de Padrões, 29 out. 2009. Disponível em: <http://www.tecmundo.com.br/seguranca/3014-inteligencia-artificial-reconhecimento-de-padroes.htm>.Acesso em: 05 fev. 2014.

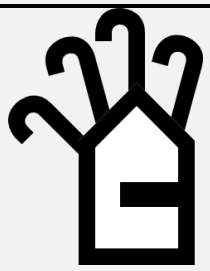
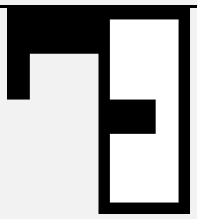
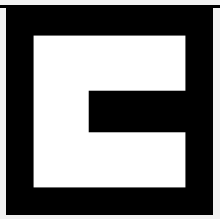
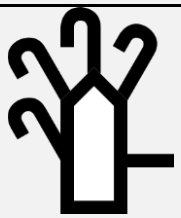
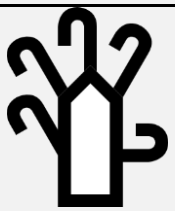
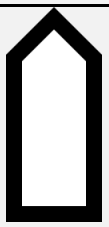
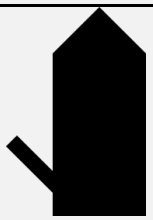
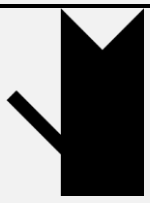
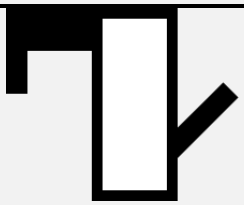
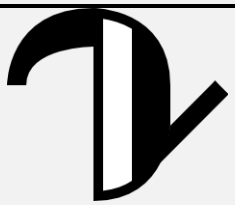
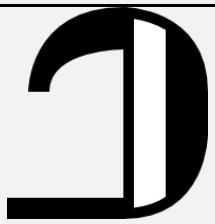
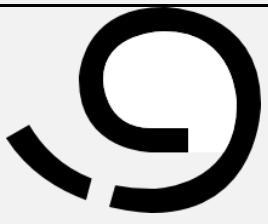
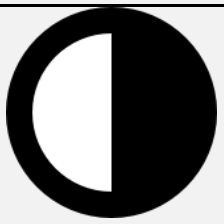
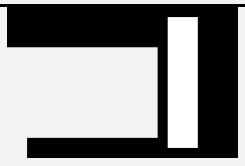
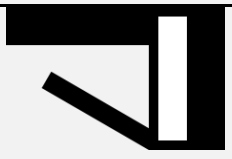
WEKA, Weka (Waikato Environment for Knowledge Analysis), Disponívelem: < http://www.cs.waikato.ac.nz/ml/weka/>. Acesso em: 12 maio. 2014.

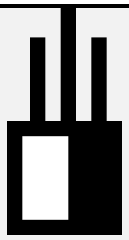
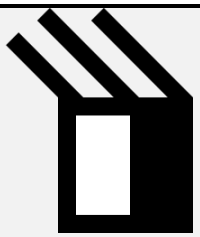
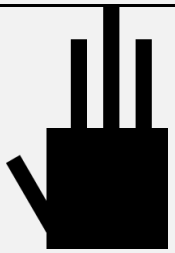
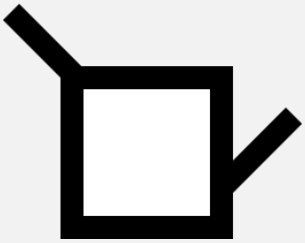
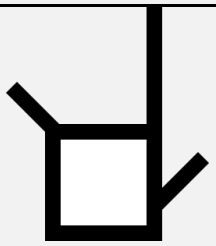
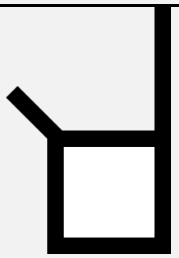
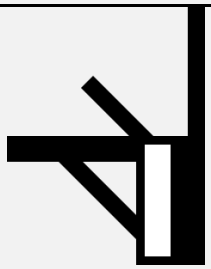
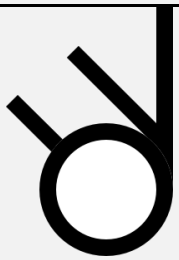
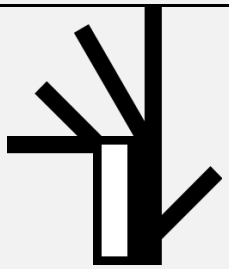
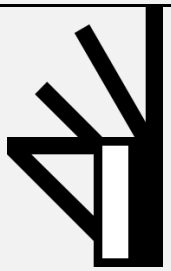
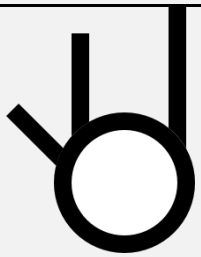
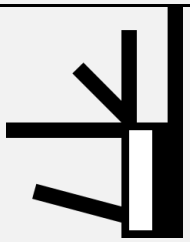
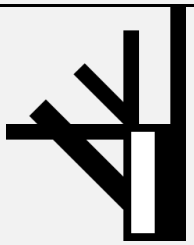
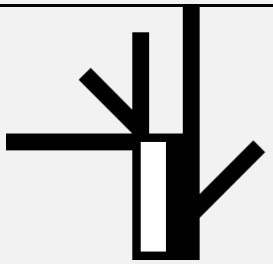
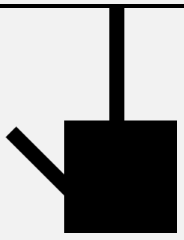
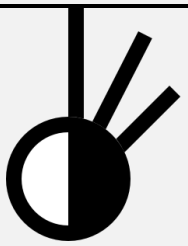
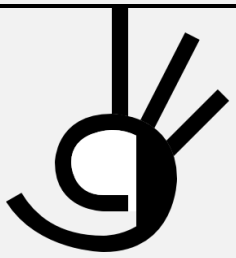
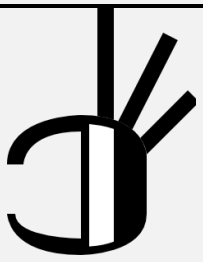
WIZBICKI, A.; BATTIST, G., **Reconhecimento de Padrões em Imagens Aplicando Visão Computacional**. Relatório Técnico-Científico. XXII Seminário de Iniciação Científica, 2014. Disponível em: <https://www.revistas.unijui.edu.br/index.php/salaoconhecimento/article/viewFile/3408/2813> .Acesso em: 10 out. 2014.

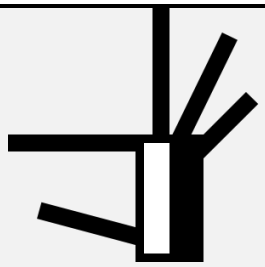
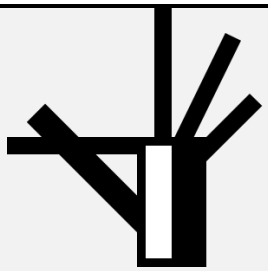
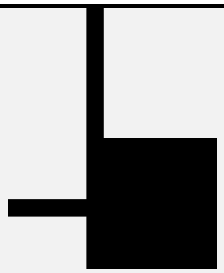
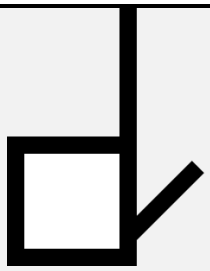
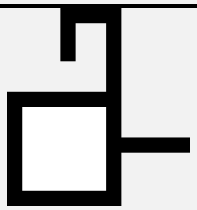
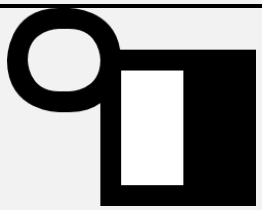
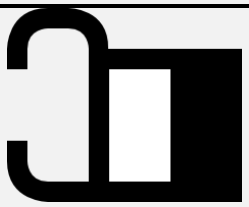
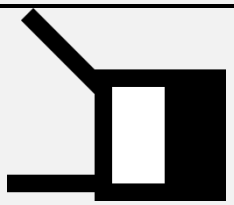
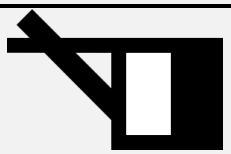
SignWriting, 2012. Disponível em: <http://www.libras.com.br/signwriting/signwriting-dp1>. Acesso em: 09 ago. 2013.

APÊNDICEA: Listagem Completa das 80 classes

 a_01	 a_02	 a_03	 a_05
 a_06	 a_07	 a_08	 a_09
 a_10	 a_11	 a_12	 a_13
 a_14	 a_17	 a_18	 a_21
 a_22	 a_23	 a_24	 a_25
			

a_26	a_27	a_28	**a_29**
 **a_30**	 **a_31**	 **a_32**	 **a_33**
 **a_34a**	 **a_34b**	 **a_35**	 **a_36**
 **a_37**	 **a_38**	 **a_40**	 **a_41**
 **a_43**	 **a_44**	 **a_46**	 **a_47**
 **a_48**	 **a_49**	 **a_50**	 **a_51**
 **a_52**	 **a_53**	 **a_54**	 **a_55**

 a_57	 a_58	 a_61	 a_62
 a_63	 a_67	 a_68	 a_71
 a_73	 a_74	 a_75	 a_76
 a_78	 a_79	 a_80	 a_82
 a_83	 a_84	 a_85	 a_86

 a_87	 a_88	 a_92	 a_93a
 a_96	 a_99	 a_100	 a_102
 a_103	 a_104	 a_105	 a_106

Fonte: (o autor)

APÊNDICE B: Saídas Weka para os dados dos Histogramas Horizontais – Base 02

1 J48

```
1   === Run information ===
2
3   Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2
4   Relation:     base2hor
5   Instances:    239
6   Attributes:   257
7   [list of attributes omitted]
8   Test mode: split 66.0% train, remainder test
9
10  === Classifier model (full training set) ===
11
12  J48 pruned tree
13  ------------------
14
15  pixel-0 <= 191
16  |   pixel-0 <= 138
17  |   |   pixel-0 <= 111
18  |   |   |   pixel-54 <= 1
19  |   |   |   |   pixel-0 <= 90: a_06 (4.0)
20  |   |   |   |   pixel-0 > 90: a_01 (5.0)
21  |   |   |   pixel-54 > 1: a_78 (4.0)
... ...
159 |   |   |   |   pixel-54 > 0
160 |   |   |   |   |   pixel-188 <= 0: a_27 (3.0)
161 |   |   |   |   |   pixel-188 > 0: a_54 (2.0/1.0)
162 |   |   |   pixel-107 > 1: a_43 (4.0/1.0)
163
164 Number of Leaves  :     75
165
166 Size of the tree :      149
167
168
169 Time taken to build model: 0.31 seconds
170
171 === Evaluation on test split ===
172 === Summary ===
173
174 Correctly Classified Instances          33               40.7407 %
175 Incorrectly Classified Instances        48               59.2593 %
176 Kappa statistic                          0.3955
177 Mean absolute error                      0.015
178 Root mean squared error                  0.1041
179 Relative absolute error                 60.6039 %
180 Root relative squared error             93.5727 %
181 Total Number of Instances               81
182
183 === Detailed Accuracy By Class ===
184
185 TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
186 1         0          1         1          1          1        a_01
187   1         0.025      0.5       1          0.667      0.987a_02
188 1         0          1         1          1          1        a_03
189  0.333     0.026      0.333     0.333      0.333      0.656     a_05
```

```
...  ...
264  0          0          0          0          0          ?          a_105
265  0          0          0          0          0          0.994      a_106
267
268  === Confusion Matrix ===
269
270  a  b  c  d  e  f  g  h  i  ... cacb<-- classified as
271    10  0  0  0  0  0  0  0  0  ...  00 |  a = a_01
272    02  0  0  0  0  0  0  0  0  ... 00 |  b = a_02
273    00  2  0  0  0  0  0  0  0  ...  00 |  c = a_03
274    02  0  1  0  0  0  0  0  0  ...  00 |  d = a_05
...    ...
349    00  0  0  0  0  0  0  0  0  ...  00 | ca = a_105
350    0  00  0  0  0  0  0  0  0  ...  00 | cb = a_106
```

Fonte: (o autor)

2 NaiveBayes

```
1  === Run information ===
2
3  Scheme:weka.classifiers.bayes.NaiveBayes
4  Relation:     base2hor
5  Instances:    239
6  Attributes:   257
7  [list of attributes omitted]
8  Test mode:split 66.0% train, remainder test
9
10 === Classifier model (full training set) ===
11
12 Naive Bayes Classifier
13
14                  Class
15 Attribute   a_01     a_02     a_03     a_05     ...    a_105    a_106
16 (0.02)    (0.02)   (0.02)   (0.02)    ...    (0.01)   (0.01)
17 ============================================  ...  ================
18 pixel-0
19 mean     99.8629   122.4 124.3429 121.2343  ...279.7714 227.3143
20 std. dev.  6.2171   0.3238   0.3238     5.44   ...0.3238   0.3238
21 weight sum     5        5        5              ...        2        2
22 precision   1.9429   1.9429   1.9429   1.9429   ...   1.9429   1.9429
23
24 pixel-1
25 mean           0        0        0        0     ...        0        0
   std. dev.   0.0017   0.0017   0.0017   0.0017   ...   0.0017   0.0017
```

```
26    weight sum    5         5         5         5         ...         2         2
27    precision     0.01      0.01      0.01      0.01      ...      0.01      0.01
28
29    pixel-2
30    mean          0         0         0         0         ...         0         0
31    std. dev.     0.0017    0.0017    0.0017    0.0017    ...    0.0017    0.0017
32    weight sum    5         5         5         5         ...         2         2
33    precision     0.01      0.01      0.01      0.01      ...      0.01      0.01
...   ...
1542  pixel-254
1543  mean    0         0         0                   ...         0         0
1544  std. dev.     0.0017    0.0017    0.0017              ...    0.0017    0.0017
1545  weight sum    5         5         5                   ...         2         2
1546  precision     0.01      0.01      0.01      0.01      ...      0.01      0.01
1547
1548  pixel-255
1549  mean          0         0         0         0         ...         0         0
1550  std. dev.    0.0017    0.0017    0.0017    0.0017    ...     0.0017    0.0017
1551  weight sum    5         5         5         5         ...         2         2
1552  precision     0.01      0.01      0.01      0.01      ...      0.01      0.01
1553
1554
1555
1556  Time taken to build model: 0.06 seconds
1557
1558  === Evaluation on test split ===
1559
1560  === Summary ===
1561  Correctly Classified Instances          39               48.1481 %
1562  Incorrectly Classified Instances        42               51.8519 %
1563  Kappa statistic                          0.4708
1564  Mean absolute error                      0.0128
1565  Root mean squared error                  0.1036
1566  Relative absolute error                 52.0554 %
1567  Root relative squared error             93.1665 %
1568  Total Number of Instances               81
1569
1570  === Detailed Accuracy By Class ===
1571
1572  TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
1573  1         0.025      0.333     1         0.5        1        a_01
```

```
1574      1          0          1          1          1          1          a_02
1575      1          0          1          1          1          1          a_03
1576      0.333      0          1          0.333      0.5        0.94       a_05
...             ...
1651 0          0          0          0          0          ?          a_105
1652      1          0          1          1          1          1          a_106
1654
1655 === Confusion Matrix ===
1656
1657 a  b  c  d  e  f  g  h  i  ...cacb<-- classified as
1658 10  0  0  0  0  0  0  0  ...  00 |  a = a_01
1659   02  0  0  0  0  0  0  0  ...  00 |  b = a_02
1660   00  2  0  0  0  0  0  0  ...  00 |  c = a_03
1661   20  0  1  0  0  0  0  0  ...00 |  d = a_05
...  ...
1736   0  00  0  0  0  0  0  0  ...  00 | ca = a_105
1737   3  0  0  0  0  0  0  0  0  ...  0  1 | cb = a_106
```

Fonte: (o autor)

3 MultiLayerPerceptron (MLP)

```
1     === Run information ===
2
3     Scheme:weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2
      -N 500 -V 0 -S 0 -E 20 -H a
4     Relation:     base2hor
5     Instances:    239
6     Attributes:   257
7     [list of attributes omitted]
8     Test mode:split 66.0% train, remainder test
9
10    === Classifier model (full training set) ===
11
12    Sigmoid Node 0
13        Inputs    Weights
14        Threshold    -0.5909650217242758
15    Node 80    -0.826749664281111
16        Node 81    -0.23739546682785256
...   ...
182       Node 247    -1.141665925516809
...       ...
13521 Sigmoid Node 79
13522     Inputs    Weights
13523     Threshold    -0.0602299962224372
13524 Node 80    -0.6406525151691054
13525     Node 81    -0.24489328000594557
...   ...
13691     Node 247    -0.5276770038216345
13692 Sigmoid Node 80
13693     Inputs    Weights
13694     Threshold    -0.007461703162840005
13695 Attrib pixel-0    0.8717836843019202
13696 Attrib pixel-1    0.03316112828854176
13697 Attrib pixel-2    -0.02884390897573923
...   ...
13949 Attrib pixel-254    0.005417494352333574
13950 Attrib pixel-255    -0.0496083955834057
...   ...
56945 Sigmoid Node 247
56946     Inputs    Weights
56947     Threshold    0.33941973797033914
56948 Attrib pixel-0    -0.8193551262394482
```

```
56949 Attrib pixel-1    0.045424127216382004
56950 Attrib pixel-2    -0.04052987521715859
 ...  ...
57202 Attrib pixel-254    -0.043901142797743
57203 Attrib pixel-255    -0.004568581028811788
57204 Class a_01
57205     Input
57206     Node 0
57207 Class a_02
57208     Input
57209     Node 1
57210 Class a_03
57211     Input
57212     Node 2
 ...  ...
57438 Class a_105
57439     Input
57440     Node 78
57441 Class a_106
57442     Input
57443     Node 79
57444
57445
57446 Time taken to build model: 539.51 seconds
57447
57448 === Evaluation on test split ===
57449 === Summary ===
57450
57451 Correctly Classified Instances          43               53.0864 %
57452 Incorrectly Classified Instances        38               46.9136 %
57453 Kappa statistic                          0.5227
57454 Mean absolute error                      0.0139
57455 Root mean squared error                  0.0907
57456 Relative absolute error                 56.141  %
57457 Root relative squared error             81.5375 %
57458 Total Number of Instances               81
57459
57460 === Detailed Accuracy By Class ===
57461
57462 TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
57463 1          0.025      0.333     1          0.5        1        a_01
```

```
57464   1          0          1          1          1          1        a_02
57465   1          0          1          1          1          1        a_03
57466  0.333       0          1          0.333      0.5        0.983a_05
 ...     ...
57541 0          0          0          0          0          ?        a_105
57542   1          0          1          1          1          1        a_106
57544
57545 === Confusion Matrix ===
57546
57547 a  b  c  d  e  f  g  h  i  j  ...cacb<-- classified as
57548 10  0  0  0  0  0  0  0  0  0  ...  00 |  a = a_01
57549   02  0  0  0  0  0  0  0  0  ...  00 |  b = a_02
57550   00  2  0  0  0  0  0  0  0  ...  00 |  c = a_03
57551   20  0  1  0  0  0  0  0  0  ...  00 |  d = a_05
 ...   ...
57626   00  0  0  0  0  0  0  0  0  ...  00 | ca = a_105
57627   00  0  0  0  0  0  0  0  0  ...  01 | cb = a_106
```

Fonte: (o autor)

APÊNDICE C: Descrição Detalhada das Saídas Weka para os dados dos Histogramas Horizontais – Base 02

1 J48

A saída do Weka para o algoritmo J48 consiste em um cabeçalho com os dados do algoritmo e da base utilizada, na impressão da árvore gerada, em um sumário com os dados obtidos, um detalhamento de cada uma das classes e uma matriz de confusão, como pode ser visto nos trechos de código abaixo.

Saída J48 – Cabeçalho

```
1  === Run information ===
2
3  Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2
4  Relation:     base2hor
5  Instances:    239
6  Attributes:   257
7  [list of attributes omitted]
8  Test mode: split 66.0% train, remainder test
```

Fonte: (o autor)

As linhas 1 a 8 correspondem a um pequeno cabeçalho do método. Nele contém o caminho percorrido no Weka até o método escolhido (linha 3), o nome da relação (linha 4), a quantidade de instâncias e de atributos (linhas 5 e 6), e o modo como foi realizado o teste (linha 8), onde 2/3 da base foram utilizados para treino (66%) e o restante para testes. Na linha onde é descrito o método escolhido (linha 3), percebe-se também a utilização de 2 parâmetros, são eles o parâmetro `C` (confidence), que corresponde ao fator de confiança inicial para a poda e por default o Weka utiliza o percentual de 25% (0.25), e o parâmetro `M`, que corresponde ao número mínimo de instâncias por folha e por default o Weka define esse número como 2.

Saída J48 – Árvore de Decisão

```
10  === Classifier model (full training set) ===
11
12  J48 pruned tree
13  ------------------
14
15  pixel-0 <= 191
16  |   pixel-0 <= 138
17  |   |   pixel-0 <= 111
18  |   |   |   pixel-54 <= 1
19  |   |   |   |   pixel-0 <= 90: a_06 (4.0)
20  |   |   |   |   pixel-0 > 90: a_01 (5.0)
21  |   |   |   pixel-54 > 1: a_78 (4.0)
... ...
159 |   |   |   |   pixel-54 > 0
160 |   |   |   |   |   pixel-188 <= 0: a_27 (3.0)
161 |   |   |   |   |   pixel-188 > 0: a_54 (2.0/1.0)
162 |   |   |   pixel-107 > 1: a_43 (4.0/1.0)
163
164 Number of Leaves  : 75
165
166 Size of the tree :     149
167
168
169 Time taken to build model: 0.31 seconds
```

Fonte: (o autor)

As linhas 12 a 166 correspondem a parte da árvore de decisão gerada pelo método, levando-se em consideração os 256 atributos e as 80 possíveis classificações. Os números entre parênteses indicam a quantidade de instâncias classificadas correta e incorretamente. Na última linha da árvore, por exemplo, para a imagem de nome `a_43` foram classificadas quatro instâncias corretamente e uma instância incorretamente. As linhas 164 e 166 indicam o número de níveis e o tamanho da árvore, respectivamente.

Saída J48 – Sumário

```
171 === Evaluation on test split ===
172 === Summary ===
173
174 Correctly Classified Instances          33               40.7407 %
175 Incorrectly Classified Instances        48               59.2593 %
176 Kappa statistic                          0.3955
177 Mean absolute error                      0.015
178 Root mean squared error                  0.1041
179 Relative absolute error                 60.6039 %
180 Root relative squared error             93.5727 %
181 Total Number of Instances               81
```

Fonte: (o autor)

As linhas 172 a 181 correspondem a um sumário com os resultados obtidos. Sendo o mais importante deles a porcentagem de instâncias classificadas correta e incorretamente (linhas 174 e 175, respectivamente).

Saída J48 – Descrição

```
183 === Detailed Accuracy By Class ===
184
185 TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
186 1          0           1          1           1          1        a_01
187   1          0.025      0.5        1          0.667      0.987a_02
188   1          0          1          1          1          1         a_03
189 0.333      0.026      0.333      0.333      0.333      0.656     a_05
... ...
264 0          0          0          0          0          ?         a_105
265 0          0          0          0          0          0.994     a_106
```

Fonte: (o autor)

As linhas 183 a 265 correspondem a uma descrição detalhada de cada classe, através da análise de precisão. Sendo o fator de precisão o mais relevante. Para a classe `a_02`, por exemplo, o índice de precisão foi de 50%, enquanto que para a classe `a_05` o índice de precisão foi de 33,3%.

Saída J48 – Matriz de Confusão

```
268 === Confusion Matrix ===
269
270   a  b  c  d  e  f  g  h  i  ... cacb<-- classified as
271   1  0  0  0  0  0  0  0  0  0  ...  0  0 |  a = a_01
272   0  2  0  0  0  0  0  0  0  0  ...  0  0 |  b = a_02
273   0  0  2  0  0  0  0  0  0  0  ...  0  0 |  c = a_03
274 02  0  1  0  0  0  0  0  0  ...  00 |  d = a_05
...   ...
349   0   00  0  0  0  0  0  0  0  ...  00 | ca = a_105
350   00  0  0  0  0  0  0  0  0 ...  00 | cb = a_106
```

Fonte: (o autor)

Por fim, as linhas 268 a 350 correspondem a matriz de confusão, que é simplesmente uma matriz quadrada que indica as classificações corretas e erradas de cada instância, onde a diagonal da matriz corresponde às classificações corretas. Para a classe `a_01`, por exemplo, o número de classificações corretas foi igual a uma, já para a classe `a_02`, esse número foi igual a dois.

2NaiveBayes

A saída do Weka para o algoritmo NaiveBayes consiste em um cabeçalho com os dados do algoritmo e da base utilizada, em um sumário com os dados obtidos, um detalhamento de cada uma das classes e uma matriz de confusão, como pode ser visto nos trechos de código abaixo.

Saída NaiveBayes – Cabeçalho

```
1  === Run information ===
2
3  Scheme:weka.classifiers.bayes.NaiveBayes
4  Relation:     base2hor
5  Instances:    239
6  Attributes:   257
7  [list of attributes omitted]
8  Test mode:split 66.0% train, remainder test
```

Fonte: (o autor)

As linhas 1 a 8 correspondem a um pequeno cabeçalho do método. Nele contém o caminho percorrido no Weka até o método escolhido (linha 3), o nome da relação (linha 4), a quantidade de instâncias e de atributos (linhas 5 e 6), e o modo como foi realizado o teste (linha 8), onde 2/3 da base foram utilizados para treino (66%) e o restante para testes.

Saída NaiveBayes – Informações Específicas do Classificador

```
=== Classifier model (full training set) ===

Naive Bayes Classifier

                  Class
Attribute   a_01     a_02     a_03     a_05     ...    a_105    a_106
            (0.02)   (0.02)   (0.02)   (0.02)   ...    (0.01)   (0.01)
==========================================  ...  ================
pixel-0
mean     99.8629   122.4   124.3429 121.2343  ...    279.7714 227.3143
std. dev.  6.2171   0.3238   0.3238     5.44   ...      0.3238   0.3238
weight sum     5        5        5              ...         2        2
precision   1.9429   1.9429   1.9429   1.9429   ...    1.9429   1.9429

pixel-1
mean           0        0        0        0     ...         0        0
std. dev.   0.0017   0.0017   0.0017   0.0017   ...    0.0017   0.0017
weight sum   5        5        5        5        ...         2        2
precision    0.01     0.01     0.01     0.01     ...      0.01     0.01

pixel-2
mean         0        0        0        0        ...         0        0
std. dev.    0.0017   0.0017   0.0017   0.0017   ...    0.0017   0.0017
weight sum   5        5        5        5        ...         2        2
precision    0.01     0.01     0.01     0.01     ...      0.01     0.01
...
pixel-254
mean         0        0        0                 ...         0        0
std. dev.    0.0017   0.0017   0.0017            ...    0.0017   0.0017
weight sum   5        5        5                 ...         2        2
precision    0.01     0.01     0.01     0.01     ...      0.01     0.01

pixel-255
mean         0        0        0        0        ...         0        0
std. dev.   0.0017   0.0017   0.0017   0.0017   ...    0.0017   0.0017
weight sum   5        5        5        5        ...         2        2
precision    0.01     0.01     0.01     0.01     ...      0.01     0.01



Time taken to build model: 0.06 seconds
```

Fonte: (o autor)

As linhas 10 a 1556contem algumas informações específicas de cada classe, ou seja, cada instância. Dentre elas podemos destacar a soma dos pesos (`weigth sum`) que corresponde ao peso total dos exemplos utilizados para estimar os parâmetros da distribuição das normas; e a precisão (`prediction`) que é o mínimo desvio padrão permitido para o atributo em questão. A precisão é obtida através de uma heurística implementada no algoritmo que calcula a média entre os valores adjacentes do atributo.

Saída NaiveBayes – Sumário

```
1558 === Evaluation on test split ===
1559
1560 === Summary ===
1561 Correctly Classified Instances          39               48.1481 %
1562 Incorrectly Classified Instances        42               51.8519 %
1563 Kappa statistic                          0.4708
1564 Mean absolute error                      0.0128
1565 Root mean squared error                  0.1036
1566 Relative absolute error                 52.0554 %
1567 Root relative squared error             93.1665 %
1568 Total Number of Instances               81
```

Fonte: (o autor)

As linhas 1560 a 1568 correspondem a um sumário com os resultados obtidos. Sendo o mais importante deles a porcentagem de instâncias classificadas correta e incorretamente (linhas 1561 e 1562, respectivamente).

Saída NaiveBayes – Descrição

```
1570 === Detailed Accuracy By Class ===
1571
1572   TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
1573 1          0.025      0.333     1          0.5         1         a_01
1574     1          0          1         1          1          1         a_02
1575     1          0          1         1          1          1         a_03
1576     0.333      0          1         0.333      0.5        0.94      a_05
 ...          ...
1651 0          0          0         0          0          ?         a_105
1652     1          0          1         1          1          1         a_106
```

Fonte: (o autor)

As linhas 1570 a 1652 correspondem a uma descrição detalhada de cada classe, através da análise de precisão. Sendo o fator de precisão o mais relevante. Para a classe `a_01`, por exemplo, o índice de precisão foi de 33,3%.

Saída NaiveBayes – Matriz de Confusão

```
1655 === Confusion Matrix ===
1656
1657   a  b  c  d  e  f  g  h  i  ... cacb<-- classified as
1658   1  0  0  0  0  0  0  0  0  ...  0  0 |  a = a_01
1659   0  2  0  0  0  0  0  0  0  ...  0  0 |  b = a_02
1660   0  0  2  0  0  0  0  0  0  ...  0  0 |  c = a_03
1661 20  0  1  0  0  0  0  0  ...  00 |  d = a_05
 ...     ...
1736   4  00  0  0  0  0  0  0 ...  00 | ca = a_105
1737 0  0  0  0  0  0  0  0  0 ...  0  1 | cb = a_106
```

Fonte: (o autor)

Por fim, as linhas 1655 a 1737 correspondem a matriz de confusão, que é simplesmente uma matriz quadrada que indica as classificações corretas e erradas de cada instância, onde a diagonal da matriz corresponde às classificações corretas. Para a classe `a_01`, por exemplo, o número de classificações corretas foi igual a uma, já para a classe `a_02`, esse número foi igual a dois.

3 MultiLayerPerceptron (MLP)

Como descrito no item 3.4.3, as redes MultiLayer Perceptron (MLP) são redes de muitas camadas. A saída do Weka para tal consiste em um cabeçalho com os dados do algoritmo e da base utilizada, na impressão da rede neural, em um sumário com os dados obtidos, um detalhamento de cada uma das classes e uma matriz de confusão, como pode ser visto nos códigos 17 a 21 abaixo.

Saída MLP – Cabeçalho

```
1  === Run information ===
2
3  Scheme:weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -
   N 500 -V 0 -S 0 -E 20 -H a
4  Relation:     base2hor
5  Instances:    239
6  Attributes:   257
7  [list of attributes omitted]
8  Test mode:split 66.0% train, remainder test
```

Fonte: (o autor)

As linhas 1 a 8 correspondem a um pequeno cabeçalho do método. Nele contém o caminho percorrido no Weka até o método escolhido (linha 3), o nome da relação (linha 4), a quantidade de instâncias e de atributos (linhas 5 e 6), e o modo como foi realizado o teste (linha 8), onde 2/3 da base foram utilizados para treino (66%) e o restante para testes.

Saída MLP – Rede Neural

```
10    === Classifier model (full training set) ===
11
12    Sigmoid Node 0
13        Inputs    Weights
14        Threshold    -0.5909650217242758
15        Node 80    -0.826749664281111
16        Node 81    -0.23739546682785256
...       ...
182       Node 247    -1.141665925516809
...       ...
13521 Sigmoid Node 79
13522     Inputs    Weights
13523     Threshold    -0.0602299962224372
13524     Node 80    -0.6406525151691054
13525     Node 81    -0.24489328000594557
...       ...
13691     Node 247    -0.5276770038216345
13692 Sigmoid Node 80
13693     Inputs    Weights
13694     Threshold    -0.007461703162840005
13695 Attrib pixel-0    0.8717836843019202
13696 Attrib pixel-1    0.03316112828854176
13697 Attrib pixel-2    -0.02884390897573923
...       ...
13949 Attrib pixel-254    0.005417494352333574
13950 Attrib pixel-255    -0.0496083955834057
...       ...
56945 Sigmoid Node 247
56946     Inputs    Weights
56947     Threshold    0.33941973797033914
56948 Attrib pixel-0    -0.8193551262394482
56949 Attrib pixel-1    0.045424127216382004
56950 Attrib pixel-2    -0.04052987521715859
...       ...
57202 Attrib pixel-254    -0.043901142797743
57203 Attrib pixel-255    -0.004568581028811788
57204 Class a_01
57205     Input
57206     Node 0
```

```
57207 Class a_02
57208     Input
57209     Node 1
57210 Class a_03
57211     Input
57212     Node 2
 ...  ...
57438 Class a_105
57439     Input
57440     Node 78
57441 Class a_106
57442     Input
57443     Node 79
57444
57445
57446 Time taken to build model: 539.51 seconds
```

Fonte: (o autor)

As linhas 10 a 57446 correspondem a própria rede neural formada. Os Sigmoide Node correspondem aos nós utilizados no backpropagation, juntamente com os dados associados a eles. Esses dados são os pesos de interconexão. Nas linhas 12 a 13691 são declarados 80 nós, ou 80 unidades, que correspondem às 80 instâncias da base. Cada um desses nós, por sua vez, possuem suas unidades ocultas, declarados nas linhas 13692 a 57203. Essas unidades possuem os pesos correspondentes aos 256 atributos da classe. Para definir o número de nós da camada oculta, o Weka, por default, utiliza uma heurística que corresponde a dividir o número de atributos somado ao número de classes por dois.

Saída MLP – Sumário

```
57448 === Evaluation on test split ===
57449 === Summary ===
57450
57451 Correctly Classified Instances          43               53.0864 %
57452 Incorrectly Classified Instances        38               46.9136 %
57453 Kappa statistic                          0.5227
57454 Mean absolute error                      0.0139
57455 Root mean squared error                  0.0907
57456 Relative absolute error                 56.141  %
57457 Root relative squared error             81.5375 %
57458 Total Number of Instances               81
```

Fonte: (o autor)

As linhas 57449 a 57458 correspondem a um sumário com os resultados obtidos. Sendo o mais importante deles a porcentagem de instâncias classificadas correta e incorretamente (linhas 57451 e 57452, respectivamente).

Saída MLP – Descrição

```
57460 === Detailed Accuracy By Class ===
57461
57462 TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
57463 1          0.025      0.333     1          0.5         1        a_01
57464   1          0          1          1          1           1        a_02
57465   1          0          1          1          1           1        a_03
57466  0.333      0          1          0.333      0.5         0.983a_05
 ...     ...
57541 0          0          0          0          0           ?        a_105
57542   1          0          1          1          1           1        a_106
```

Fonte: (o autor)

As linhas 57460 a 57542 correspondem a uma descrição detalhada de cada classe, através da análise de precisão. Sendo o fator de precisão o mais relevante. Para a classe `a_01`, por exemplo, o índice de precisão foi de 33,3%.

Saída MLP – Matriz de Confusão

```
57545 === Confusion Matrix ===
57546
57547 a  b  c  d  e  f  g  h  i  j  ... cacb<-- classified as
57548   10  0  0  0  0  0  0  0  0  ...  00 |  a = a_01
57549   02  0  0  0  0  0  0  0  0  ...  00 |  b = a_02
57550   00  2  0  0  0  0  0  0  0  ...  00 |  c = a_03
57551   20  0  1  0  0  0  0  0  0  ...  00 |  d = a_05
 ...     ...
57626   00  0  0  0  0  0  0  0  0  ...  00 | ca = a_105
57627   00  0  0  0  0  0  0  0  0  ...  01 | cb = a_106
```

Fonte: (o autor)

Por fim, as linhas 57545 a 57627 correspondem a matriz de confusão, que é simplesmente uma matriz quadrada que indica as classificações corretas e erradas de cada instância, onde a diagonal da matriz corresponde às classificações corretas. Para a classe `a_01`, por exemplo, o número de classificações corretas foi igual a uma, já para a classe `a_02`, esse número foi igual a dois.